The Small Computer Magazine

# kilobaud

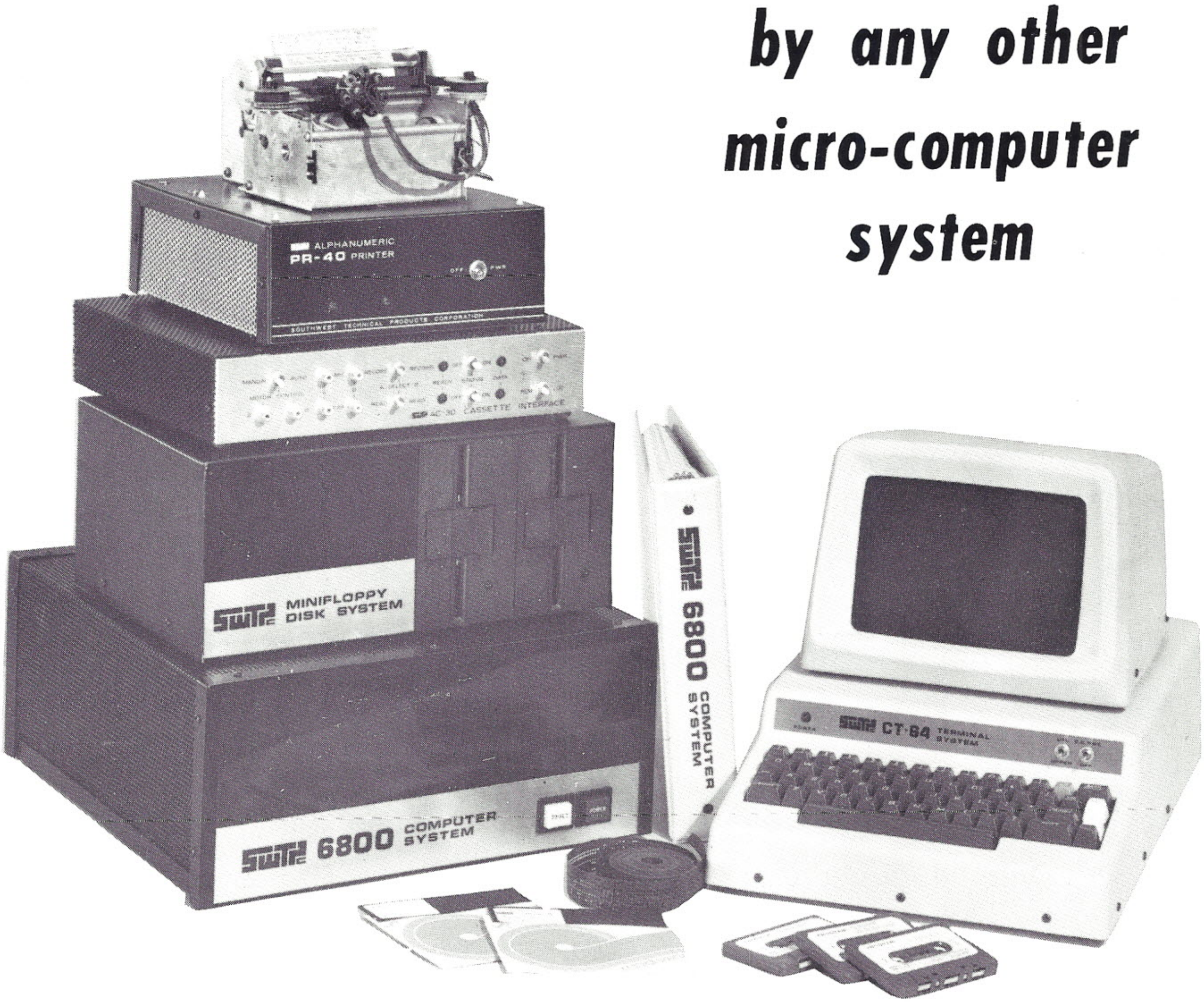
T.M.

Understandable for beginners . . . interesting for experts

*March 1978 / Issue #15 / $2.00 / DM7,50 / Sfr8,10 / Ffr 16,0*

# VERSATILITY – UNMATCHED

## *by any other micro-computer system*




| | | |
|---|---|---|
| COMPUTERS | TAPE SYSTEMS | MEMORY – 4K, 8K, 16K & 32K |
| TERMINALS | DISK SYSTEMS | PROGRAMMABLE REAL TIME CLOCK |
| PRINTERS | GRAPHICS | CALCULATOR INTERFACE |
| SOFTWARE | PROM PROGRAMMER | A - D CONVERTER |

Write, or circle our reader reply number for a catalog describing our complete line of computer hardware and software.

NOW AVAILABLE IN BOTH KIT AND ASSEMBLED FORM.

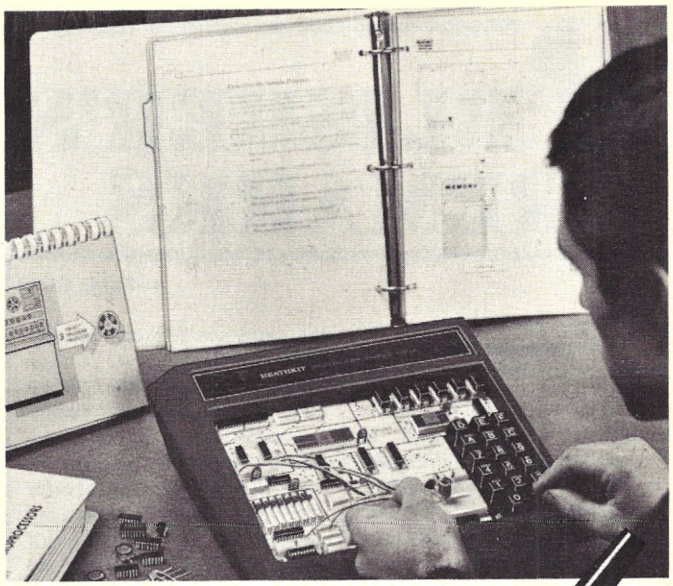
SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 S6

# LEARN Microprocessor Operation, Interfacing and Programming with

# The New HEATHKIT Microprocessor Course!

*Here's the knowledge you need to really use your computer's power!*




The more you learn about computers, the more you'll appreciate how important it is to fully understand microprocessor operation and techniques — and here's the easiest, fastest and lowest-cost way to do it! Using Heath's proven self-instruction techniques, the Course gives the knowledge you need to expand your computer interests wisely — and to fully utilize the immense power your own personal computer puts at your fingertips.

Working with the popular 6800 microprocessor, you'll explore this exciting field in 3 easy steps! In step 1, using proven self-instruction text materials, you'll learn about micro-computer basics, number systems and codes, computer arithmetic, programming, 6800 capabilities and interfacing and more. In step 2, with the aid of audio visuals included, you'll go deeper into programming, designing with microprocessors and semiconductor memories. Then, in step 3, you'll actually perform 19 "hands on" experiments with the optional trainer designed to reinforce the theory you've mastered to that point!

The ET-3400 Trainer kit, designed to accompany your course, features the popular 6800 microprocessor. It is actually a miniature digital computer in itself, complete with a: 1K ROM monitor program; 6 digit LED display for address and data readout; 17-key hexadecimal keyboard for entering programs, data and control; 256 bytes of RAM (expandable to 512 bytes with the chips supplied in the course); breadboarding socket; 8 buffered LED's for display of logic states; 8 SPST switches for binary input plus an on-board power supply with +5, +12 and −12 volt outputs.

The Course comes complete with 62 electronic components, including a 6280 PIA, two 2112 RAM's, a 1406 digital-to-analog converter, 741 and 301 op amps and more. Includes audio visual aids such as an audio cassette, colorful flip charts and programmed learning text material in two deluxe permanent binders.

Complete the optional exam and receive a Certificate of Achievement and 8.0 Continuing Education Units (CEU's) — a nationally accepted means of recognizing participation in non-credit adult education. (Note: Microprocessor Course requires completion of Digital Techniques Course or equivalent knowledge.)

ORDER ETS-3400
(Course and Trainer kit) .................... **$269.95**
(Add $3.06 shipping and handling)

## FREE SOLDERING IRON!

**Included with your Course at no extra cost — this pencil-style soldering iron that's perfect for printed circuit work — a $7.95 comparable value — yours FREE!**

## MONEY-BACK GUARANTEE!

If you are not 100% satisfied with your Heathkit Course, we will refund the full purchase price of the course text material, less trainer.

## NEED DIGITAL TECHNIQUES TRAINING?

**Comprehensive Digital Techniques Course** — the background you need to understand computer and modern electronics technology! Digital fundamentals, semi-conductor devices for digital applications, Boolean algebra, flip-flops and registers, sequential and combinational logic circuits, digital design and an introduction to computers and more! The course includes programmed instruction text, audio-visual aids, parts for experiments and more.

Digital Design Experimenter/Trainer lets you perform all the experiments in your course, then develop and test your own projects.

Features solderless breadboard sockets, 4 binary data switches, 2 "no bounce" switches to pulse logic circuits, 3-frequency pulse clock generators, 4 LED's for visual indication of logic states plus 3 regulated power supplies. Upon completion of the optional final exam, you receive a Certificate of Achievement and 4.0 Continuing Education Units.

ORDER EES-3201 (Course and Trainer Kit) .............. **$119.95**
(Add $3.64 shipping and handling)

## SAVE TIME--ORDER BY PHONE Call (616) 982-3411

HEATH
Schlumberger

**Heath Company, Dept. 351-392**
**Benton Harbor, Michigan 49022**

H5

## OR, MAIL COUPON TODAY!

Gentlemen: Please send me the item(s) checked below and include my FREE Soldering Iron.

☐ Microprocessor Course and Trainer (ETS-3400) $269.95 (+$3.06 shipping & handling)

☐ Microprocessor Course only (EE-3401) .........$89.95 (+$2.23 shipping & handling)

☐ Microprocessor Trainer kit only (ET-3400) ......$189.95 (+$1.69 shipping & handling)

☐ Digital Techniques Course and Trainer kit (EES-3201) ....................................$119.95 (+$3.64 shipping & handling)

Prices subject to change without notice

I enclose ☐ check ☐ money order for $________; OR, (Michigan residents, add 4% sales tax)

CHARGE to my ☐ VISA/BankAmericard ☐ Master Charge

Account #________________ Exp. Date________

Master Charge Code #________________

Signature________________________
Necessary to send merchandise

NAME________________________
Please print

ADDRESS________________________

CITY________________STATE________ZIP________

# PUBLISHER'S REMARKS

*Wayne Green*

### Irrelevant Ads

One of our readers sent in a nice letter with a long list of reasons why he liked *Kilobaud*. I could read letters like that all day. But then he went on to mention that he appreciated our not running ads for items not related to computing . . . and I reached for my typewriter to try to put this into perspective.

Let's say that you are the publisher of *Kilobaud*. It's fun at times, but it can be very aggravating. Let's suppose that your main goal is to attempt to bring as much fun and education to your readers as possible, and that you've long ago given up on ever making a lot of money in publishing. Your main goal in life is to not get so far into hock that everything falls in on you. You've grown accustomed to watching others worry about managing their wealth while you manage your poverty.

You know that the fundamental rule of thumb in publishing is that advertising income must pay the printing bill. This means that the more ads you can get, the more articles you can publish. The formula is about 40 percent advertising and 60 percent editorial material for a break-even system. Thus, every time you sell a full-page ad, whether it be for dog food or floppy disks, you can then publish a page-and-a-half article. Would you, as a publisher, voluntarily cut down on the number of article pages if you didn't have to? It's something to think about.

We do have a policy of not accepting ads from firms with which we have had trouble or about which we've received reports from readers of troubles. This costs us a few pages of ads each month.

About the only ads not related to computers we've had so far have been for windjammer cruises . . . and these are trade-offs that we use as prizes in our subscription contests and as prizes for advertisers in some ad contests. The cruises are great and make fine prizes; I wish I could get enough time to check one out myself. I love to skin-dive and visit those out-of-the-way islands in the Caribbean. Well, perhaps this year I'll make it.

We haven't made any effort to sell car, liquor and cigarette ads in *Kilobaud* . . . and we haven't had any; but such irrelevant ads would make it possible to publish more good articles than we do now. Would you like to see five or ten more interesting articles each month?

Mind you, I have nothing at all against running all of the reputable microcomputer manufacturing and store ads we can get. I'd love to have *Kilobaud* thicker than *73*, which is running over 250 pages most months. You can help us get the industry to support *Kilobaud* by making sure that you use the reader's service card in the back of the magazine every month. You can also give *Kilobaud* a little boost if you write to any non-advertisers.

The way this field is growing, I think we could come up with several hundred pages of articles every month . . . if we could get the advertising support to pay the printing bills.

### Send Card—Get Life

Inside the back cover of each issue of *Kilobaud* is a page with three cards on it. One of these contains numbers keyed to advertisements about which you can get further information. Just circle the ad-codes that interest you, tear out the card and mail it in. Dozens of readers have *not* been sending in cards.

Henceforth, until further notice (or until we forget about it), we will put all of these cards into a box each month and hold a drawing. The winning card, chosen at random, will result in a lifetime subscription to *Kilobaud* for the sender. With this as a possible prize, perhaps your subconscious will drive you to tear out a card, mark it with appropriate circles and send it in.

Most of the advertisers are sitting on the edge of their chairs, waiting for word that you would like to see their spec sheets. Make them happy by asking . . . and you just might win a lifetime subscription to *Kilobaud* to boot.

### Writing Better Ads

The basics of selling are basics, and it doesn't make much difference whether you are selling through ads in a magazine, on television or in a store.

But, for some reason, very few manufacturers take the time to read any books on how to write ads. Even worse, the technical nature of the computer throws many ad agencies off their stride and they start listening to manufacturers instead of using their own knowledge of selling. The result of this is that there are few ads in the magazines that could not be improved enormously. Many ads could be redone to at least double sales.

# EDITOR'S REMARKS

*John Craig*

*Gordon French is one of the earliest pioneers in the field of personal computing . . . and a longtime friend of mine. The very first computer club in this country (the Home-brew Club) was started in Gordon's garage way back in March 1975 in Menlo Park CA. He recently mentioned that he plans to run for the presidency of the International Computer Society. It occurred to me that perhaps Gordon would like to write a guest editorial for* Kilobaud *and discuss his feelings about the ICS and where he would like to see it go under his leadership. He accepted the invitation. I hope you'll give him your support in this effort because I'd like to see the ICS become the organization it should be.—John.*

It is obvious to those of us who have been in the computer business that the motion of the market for computers from industry to the home is going to change, for all time, the atmosphere and the environment of our homes, just as it has changed our industry. It is also clear to those who study history that we will find it almost impossible to foretell with any accuracy most of the changes that this shift will bring about. Just as the automobile has become a part of the world scene, so the computer will inevitably change the world as we perceive it today. Those who anticipate that the computer in the home is going to solve problems rather than create new ones have a narrow view of history.

### Automobile/Computer—Analogies

When the first auto clubs came into existence, drivers' licenses and auto insurance were not great issues. The common interest of drivers in decent roads, adequate service and handy access to adequate fuel caused them to form clubs to bring pressure to bear on legislatures and manufacturers. Only the recognition that they, acting in concert, could do what needed to be done brought them to expend money and effort to secure what they could not obtain alone. But they never anticipated what really occurred. Through concentrated effort, they could publish road maps, tours; suggest preferred places for food, lodging and service; secure beneficial insurance rates; warn each other of traps and pitfalls, and influence legislation and business procedures.

In computing, we are at that precise point now.

### Technology Marches On—Sort of

The personal computer *will* make our world a different place.

However, we cannot now determine just what that place will be like. Could anyone trading the "old gray mare" for a Tin Lizzie have anticipated drive-up tellers, drive-in theaters or recreational vehicles? With imagination, some could foresee the need for the interstate highways, recaps and seat covers. Those same types are now developing software on inexpensive cassettes and modems so that we can talk to one another.

But neither American Telephone and Telegraph nor the federal government has yet taken much cognizance of the personal computer. No national broadcasting network has yet contemplated a "Computer Game of the Week," apparently unaware that many of us are watching a different kind of tube many of these evenings.

The phenomenon of a nation whose technological prowess has enabled the microcircuit to be mass produced has now made a universally useful device to put the screwdriver, paperclip and rubber band to shame. I am no more of a prophet than anyone else, but I can guess that the real reason people are buying and using Citizens Band radio, for example, is that they can once again be mutually friendly and "talkative" without involvement and without risk of contracting communicable disease. What might happen if they discover they can engage with others in games, exchanges and mutual entertainment?

One thing is clear and easy to see: We do need a special-interest group composed of people like ourselves—a group large enough to be able to effectively look after the interests of the personal computer user. We need a group with enough collective clout to be heard in senate chambers and boardrooms. Who is going to tell insurance companies that a personal computer is a private belonging—as is a camera or a hi-fi—and should be similarly insured?

### There Is Such a Group

The International Computer Society was originally formed as the Southern California Computer Society (SCCS). What is now *Interface Age* started as the journal of the SCCS. Early growing pains of both the SCCS and Bob Jones' magazine split the magazine into a private venture, while the SCCS contended with monthly meetings where 2000 people made an all-day affair of getting together to swap equipment, software and experiences.

Chapters of the SCCS were formed all over California, and hopes ran high that the SCCS would reach all over the United States. In fact, it happened.

Too much of it happened—in fact too little of it was planned! All-volunteer laborers, from board members to coffee pourers, were faced with coping with the extraordinary and sudden interest in personal computers. People from all over the world joined the SCCS and enthusiasm ran high.

The job of putting out a small, slick magazine on behalf of the SCCS soon began to require more of Bob Jones' time and, later, more of his money. Meanwhile, with letters to answer and meetings to be arranged on a grand scale, the board failed repeatedly to resolve and conclude a sound business arrangement with Jones. With a growing subscription list and advertisers anxious to buy space, and with his own money at stake, Jones failed repeatedly to resolve and conclude a sound business arrangement with the SCCS. The result was that the SCCS, with 8000 members worldwide, did not have a journal.

The rumors and opinions about who should have done what have, I believe, finally subsided. It appears that an amicable settlement will soon be reached. I have talked with Bob Jones as well as with some members of the board; and everyone agrees that settlement can, should and will be reached. I agree.

Though the magazine has not been the SCCS's only problem, it certainly is the principal reason that the personal computing public is not aware of its existence.

Last year, SCCS officially changed its name to the International Computer Society.

If you doubt the need of an international computer society, consider the manufacturers who refuse to standardize their versions of BASIC. What about the upcoming problems in standardizing hardware, software and operating procedures for personal computer networks? Who else is going to deal with legislatures, Ma Bell and the FCC? Only if we join together are we going to have the clout necessary to deal with the issues and such organizations. We cannot do it as small local clubs. The International Computer Society is a place where we can all come together.

The ICS now publishes its own journal, *Microcomputer Interface,* in a slick magazine format that carries advertisements, editorials, software and hardware reviews. I get it for Phil Feldman and Tom Rugg's software, which makes it well worth the price. Fifteen dollars a year covers not only the magazine, but also membership for a year in the International Computer Society.

For those who want to join the Society, the address is: International Computer Society, Box 54751, Los Angeles CA 90054.

Make your $15.00 check payable to the Society.

### My Own Horn

I am running for the presidency of the Society for the coming year. I have owned computers since 1974. The first was an 8008 system (I still have it and it still runs). Fred Moore and I started the Home-brew Computer Club in March 1975; I am still club librarian and a director of the club.

By December 1975, I was time-sharing a desk with three other guys in a hole in the wall in Berkeley. A big sign out front read, "Processor Technology Corporation." I was project engineer for an integrated microcomputer that for some strange reason was named SOL. I directed the effort to design and build the sheet metal and wrote the original sheet-metal-assembly instruction manual.

In spring 1977, my longtime associate, Gregory Yob, and I presented a paper on home computer interfaces to the IEEE. I started the North Star users' group in my home and set up its library.

I am not to be confused with John French, who heads Alpha-Micro Systems, nor with Don French, who knows Charles Tandy. Although we are not related, we Frenches call each other "cousins."

This fall, I began to work for Commodore Business Machines as manager of customer applications software.

I view the home computer as an expansion and an extension of human capabilities. I think that the International Computer Society will fill a need for all of us who are integrating this new tool into our environment and our society. I welcome your support—of both the Society and of myself as president.

**Gordon French**
**Menlo Park CA**

kilobaud

**Publisher**
Wayne Green
**Executive Vice President**
Sherry Smythe
**Editor**
John Craig
**Managing Editor**
John Barry
**Editorial Assistants**
Dennis Brisson
Susan Gross
**Production Department**
Lynn Panciera-Fraser
Craig Brown
Gayle Cabana
Robert Drew
Michael Murphy
Weston Parker
Bob Sawyer
Noel R. Self
Robin M. Sloan
**Typesetting**
Barbara J. Latti
Pauline Halvonik
Sharon Horton
Jennifer Johansson
Mary Kinzel
Marie Walz
**Photography**
Bill Heydolph
Tedd Cluff
**Associate Editors**
Don Alexander
Tim Barry
Sheila Clarke
Rich Didday
Phil Feldman
Tom Rugg
Peter Stark
**Bookkeeper**
Knud E. M. Keller
**Assistant Bookkeepers**
Nancy Dupuis
Frances Marion
Dominique Smith
Cida Teixeira
**Marketing**
Sherry Smythe
Jack Shimek
Cynthia Gray
**Circulation**
Dorothy Gibson
Barbara Block
Fran Dillon
Florence Goldman
Valerie Horn
Rhonda Ramsey
**Receptionist**
Doni-Anne Jarvis
**Computer Data Control**
Judy Waterman
Judy Brumaghim
Linda Cate
**Computer Programming**
Ron Cooke
Richard Dykema
Steven Lionel
**Service Consultants**
Microcomputers Inc.
**Printing**
Michael Potter
William Cering
Dwight Perry
**Mailroom**
Theresa Toussaint
Sue Chandler
Ethan Perry
Maureen Toussaint
**Plant Maintenance**
Bill Barry
Ken Cox
Lorraine Pickering
**Advertising**
Mary Jo Sponseller
Heidi Kulish
Marcia Stone
**European Distributor**
Monika Nedela

Kilobaud is published monthly by 1001001, Inc., Peterborough NH 03458. Subscription rates in the U.S. and Canada are $15 for one year and $36 for three years. In Europe: Kilobaud erscheint montatlich bei Fachzeitschriftenvertrieb Monika Nedela, 7778 Markdorf, Markstr. 3. Abonnement DM 70.- plus Porto 7.20 + gesetzl. MWST; Sfr. 81.- plus Porto 7.20. Please write for other foreign rates. Second-class postage paid at Peterborough NH 03458 and at additional mailing offices. Publication No. 346690. Phone: 603-924-3873.

INCLUDE OLD ADDRESS AND ZIP CODE WITH ADDRESS CHANGE NOTIFICATION.

# BOOKS

*Advanced BASIC Applications and Problems*
**James S. Coan**
**Hayden Book Co., 1977**
**184 pages, $7.95**

This paperback should be at the top of your must-get book list!

*Kilobaud* No. 9 (September 1977) contained a review by John Arnold of *Basic BASIC; An Introduction to Computer Programming*, authored by Mr. Coan. That book was good, and Mr. Coan's latest, *Advanced BASIC*, is excellent.

Let's take a look at some of the information packed into this book.

A brief review of BASIC is presented in the initial chapter. For the experienced BASIC programmer, a quick glance through this chapter should be sufficient. The novice, on the other hand, should reread *Basic BASIC* and then continue into Chapter 1 of *Advanced BASIC*. Some of the highlights in this first chapter include looping, arithmetic operations, conditional operators, arrays and subroutines. Chapter 2, entitled "Extended Features of BASIC," includes the TAB statement, PRINT USING, etc. Although the program listings throughout the book were run on a GE or HP system under a time-sharing BASIC, most, if not all, of the important features of these BASIC systems are included in a reasonably sophisticated microcomputer system.

Strings are introduced in Chapter 3, with plenty of program listings and explanations covering the string array, the subscripted variable and the substring.

Since a GE or HP system was used to verify the programs given in the book, Chapter 4 (Files) relies on the characteristics of these particular systems. However, what micro does not include READ and DATA statements in its BASIC software? Again, it should be no problem running the programs on a typical 8K BASIC supplied by most microcomputer manufacturers.

For the graphics people out there, Chapter 5 is devoted to plotting on a Teletype or similar TV-type screen device. Some really good programs are included in the pages of this chapter.

Chapters 6 through 11 are mathematically oriented and filled with program listings and explanations needed to find: the area under a curve; points in a plane; parallel and perpendicular lines; and answers to other coordinate-geometry problems.

An entire chapter is devoted to polynomials. Sequences and series cover yet another chapter . . . all this and program listings to go along with a fine explanation of the problems and possible solutions.

Simultaneous linear equations are covered in great depth; the chapters on mathematical problems are well presented. A problem is stated, a possible solution is given in the form of a program listing, then the serious reader is given the chance to improve the original solution or increase the complexity of the problem.

Statistics is given its fair shake, also. Do you need quick routines to find averages, variances, standard deviations, medians? Chapter 11 has them. Finally, simulation and games are covered in the final chapter. The emphasis, again, is on problem/solution techniques.

Not only does this book contain hundreds of BASIC program listings, but each section is summarized and problems are presented for the serious reader to solve by using the information and techniques previously presented within each particular section. So, for the price, you get a tutorial on a particular aspect of BASIC programming, accompanied by a program listing, a good fundamental insight into a particular problem and a possible solutioning method to the problem.

Many of the problems at the end of each section are solved (i.e., a program listing is given) in the back of the book.

All in all, this is an excellent book to have handy, not only on your bookshelf for future reference, but also at your elbow the next time you sit down at your micro.

**Len Gorney**
**Clarks Summit PA**

*Computer Algorithms and Flowcharting*
**Silver and Silver**
**McGraw-Hill, 170 pages**
**$8.55, Softbound**

If you are looking for instructions in programming, written in plain English, devoid of "computerese," this book is for you.

Unlike most other writers on the subject of programming, authors Gerald and Joan Silver seek to impress their student readers rather than their peers in the teaching profession. This difference in emphasis produces a bonus for the novice programmer.

An "algorithm," as defined by the Silvers, is ". . . a precise set of well-defined rules or procedures for the solution of a problem."

A flowchart is simply a visual outline of an algorithm.

The authors claim that time spent in planning a programming project represents an investment in the future.

Anyone who has tried to create a workable program by typing a vaguely conceived set of commands into his computer's keyboard can appreciate the need for developing a systematic approach to problem solving.

The authors view the subject of programming as being composed of six elements: (1) analyzing the problem; (2) developing algorithms and flowcharts; (3) coding; (4) keyboarding; (5) testing and debugging; and (6) documenting. They devote most of their attention to explaining how the novice can meet the challenge presented by the first two elements.

Besides covering elementary programming theory and flowcharting, the book lays a foundation for the reader's understanding of more complicated (sophisticated) programming techniques.

American National Standard Institute (ANSI) flowcharting symbols are used throughout. These consist of circles, squares, rectangles, parallelograms, etc., interconnected by lines to indicate the path that a computer must follow as it solves a specific problem.

Solutions to real problems are presented to demonstrate the use of programming techniques such as branches, loops, counters and arrays. Solutions are presented in three computer languages: BASIC, FORTRAN and COBOL. Such side-by-side solutions provide an unparalleled opportunity for the newcomer to programming to become acquainted with unfamiliar instruction sets and to appreciate the strengths and weaknesses of each.

Almost half of the book is devoted to flowcharting illustrations.

If you have ever had to leaf through pages, forward or backward, in pursuit of an illustration referenced in text material, you are certain to appreciate the layout of this book. The authors and editors have made an obvious—and successful—effort to keep textual explanations and their supporting illustrations on the same page or on an opposing page.

The book's layout reinforces the authors' contention that detailed planning solves problems—for readers as well as for computers.

**Sherman P. Wantz**
**Sebring FL**

*EDN Magazine, 11/20/77*
**Cahners Publishing**
**Boston MA 02116**
**$4 (Special Issue), 313 pages**

You've never seen an issue of a *magazine* reviewed in the book-review section of *Kilobaud*, and it's doubtful you ever will again. But, I recently finished reading the November 20th, 1977, issue of *EDN* . . . and I have to share it with you!

*EDN Magazine* is published "exclusively for designers and design managers in electronics," but they certainly won't balk at selling you a copy of this issue for $4. The *Special Issue* stamped on the front cover is really an understatement. It's a *Fantastic Issue* and is chock-full of reference material and features that will be of interest to anyone involved in microcomputers (either professionally or as a hobby).

I want to tell you about the best part first (since I can hardly contain my enthusiasm for it): a seven-chapter series entitled "EDN System Design Project," written by *EDN*'s computer editor, John Conway (and other members of the staff). It describes how, as the result of a bet, they designed, and built, a truly

low-cost microcomputer system for small business and scientific applications . . . for under $3000.

The project, called Indecomp, initially came about following a discussion they got into with the engineering manager of a large minicomputer firm about what exactly constituted a *low-cost* system. (The mini manufacturer was offering a "low-cost" system ranging from $30,000 to $150,000!) They argued that their definition of a low-cost system was one that could be built for five to ten thousand—at the very most. The engineering manager replied, "If you can build a practical business system for less than $10,000, I'll personally give you that amount." They didn't plan to collect from the gentleman, but the challenge was accepted. A $3000 budget was allotted for the project; and the trials and tribulations of the effort (Chapter 7) make for interesting reading.

Guess what single-board computer they decided on (after much research)? It was the Apple-II. (You could say a lot about their good judgement . . . and the Apple-II.) The system was installed in a secretary's inexpensive desk and consisted of the Apple, a video monitor and keyboard, power supply, Expandor printer and dual Phi-decks. The software consisted of scientific and business programs extracted from Osborne's *Some Common BASIC Programs.* The final price tag was just over $3000. Though I'm not convinced the software requirements for a practical system have been met, the entire effort is worthwhile; anyone contemplating building a system from scratch (using a single-board computer) will find it interesting indeed.

The first six chapters in the System Design series cover CPU board offerings, different bus structures, software, memories, interfacing, terminals . . . and more. Other sections of this issue contain goodies such as *EDN*'s "4th Annual Microprocessor Directory" (recipe-type "cards" that contain specs and block diagrams on every microprocessor currently being manufactured). I've made reference to the "3rd Annual Microprocessor Directory" on many occasions during the last year.

The "1st Annual Microcomputer Support Chip Directory" makes its debut in this issue and is a valuable addition, but that's not all. To top it all off, this issue contains the "3rd Annual Microcomputer Systems Directory," which lists the features, prices and configurations of *all* the commercial and hobbyists systems available. (Oops, almost forgot something: there is also a fine article by Carol Ogdin on programming languages.)

You won't find another source of reference material of this quality and quantity *for only $4.* If I've convinced you, then let me suggest you rush off a check or money order for $4 (Massachusetts residents, add 5 percent sales tax) to: EDN Microcomputer Reprints, 221 Columbus Ave., Boston MA 02116. Make it payable to EDN Reprints.

**John Craig**
**Editor**

# NEW PRODUCTS

## New Logic Analyzers from Paratronics

The Model 100A Logic State Analyzer ($295) is designed to operate as an 8-channel stand-alone analyzer or with the identically priced Model 10 expander unit, which adds 16 inputs and several advanced features.

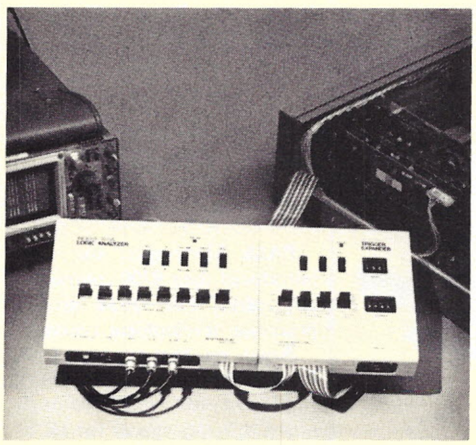
In the stand-alone mode, the Model 100A offers a 16-word truth table display of ones and zeros using any ordinary oscilloscope; combinatorial logic triggering; post-trigger and pre-trigger data collection; hex and octal formats; and both snapshot and repetitive display presentations.

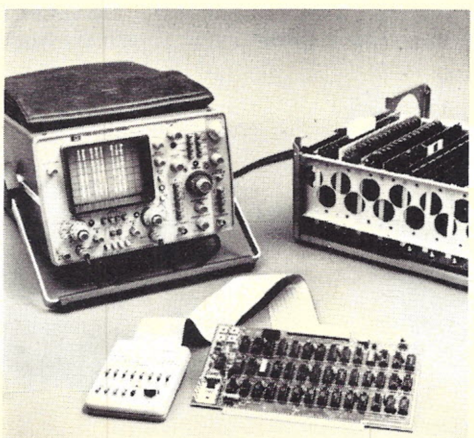
The Model 100A can be mechanically mated with the Model 10 on an optional baseplate to provide a fully integrated 24-bit logic analyzer package. The Model 100A/10 package also provides a user-programmable digital delay for paging through programs up to 1000 steps long and a pass counter to permit the microprocessor's state to be monitored after "n" passes through a loop. For capturing and displaying selected bus operations, the Model 100A/10 provides the required clock and trigger qualifiers as well. Both models can be used directly with a variety of logic families, including TTL, Schottky, MOS, CMOS and DTL.

Both units include 100-page applications manual, and both are available in assembled or kit form. Model 100A and Model 10 assembled, $295 each; Model 100A and Model 10 kit, $229 each; optional baseplate $9.95; separate owner's manual, $4.95 each.

Paratronics has also developed the Model 150 "Bus Grabber" logic analyzer, a one-board package that electrically and mechanically interfaces to the popular S-100 bus. ($369 kit; $449 assembled.)

Paratronics, Inc., 800 Charcot Avenue, San Jose CA 95131.


*Model 100A and Model 10 mated.*


*Model 150 "Bus Grabber."*

## 6502 Program Exchange

The 6502 Program Exchange has released new software packages for 6502 systems, including an extended version of the high-level language FOCAL, a 4K resident assembler and an efficient Mini-Editor.

The new FOCAL is called FCL65E to distinguish it from the previously released FCL-65. FCL65E (6.5K) offers 8-9 digit accuracy, 8-level priority interrupt handling, string variables and functions, and greater flexibility in its FOR, SET and DO commands. Complete cross-assembly listings for TIM ($1000-$25F2) and KIM ($2000-$35F2) cost $35. Both FCL-65 and FCL65E now have all their system dependent software in a zero-page I/O block, allowing easy conversion to other 6502 systems.

A Mini-Manual ($6) and a paper tape or hex dump ($17) will get you started on TIM or KIM systems. A user's manual, 104 pages of FCL65E examples and further documentation is available for $12. The Exchange offers an expanding library of programs for FCL-65 and FCL65E.

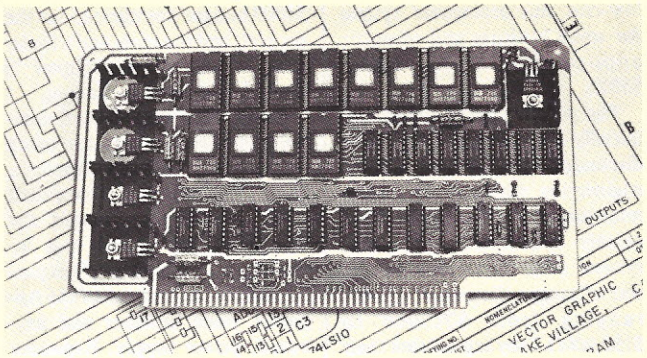
More information and a list of other available software may be obtained for $1.

The 6502 Program Exchange, 2920 Moana, Reno NV 89509.

## Pertec's New Attaché

A compact desk-top computer has been introduced by Pertec Computer Corporation's Microsystems Division, 2111 Erwin St., Woodland Hills CA 91367. Called the Attaché, the

*Attaché desk-top computer.*




*Vector Graphic PROM/RAM board.*

25-lb. unit, built around the 8080 MPU, is available through more than 40 Mits dealers across the nation. Its circuitry uses the S-100 bus configuration with a 10-slot board capability.

Standard features include LED indicators for on/off and systems status; a reset switch that returns to PROM monitor, which controls operation of the computer from the keyboard; and a video output jack (75 Ohms). The video output provides full uppercase and lowercase character generation, 16 lines of 64 characters and a choice of black on white or white on black character display. A 1K RAM with extra sockets added for PROMs on the turnkey board is standard.

Available options include an audio cassette recorder (KCACR) board; floppy-disk systems and software; memory boards; a 16K ROM BASIC board; and C Save and C Load cassette routines (included in BASIC).

Keyboard, CPU board, video board and turnkey monitor board are provided in the basic configuration, which retails for $1,449, assembled and tested.

**New Noval 760 Computer**

Noval, Inc., of San Diego CA has updated its 760 series computer system to allow rapid and efficient software development. The interaction between the editor and assembler allows the user to edit, assemble and debug applications programs without having to externally save or reload source or object code. The self-contained 760 now incorporates a Z-80 microprocessor, a full 32K user memory (RAM) plus an additional 1K scratchpad and 1K video refresh memory. The unit features a fully programmable character generator (2K) and a 3K of system utility routines on PROM. Also included are a professional 12 inch TV monitor, digital cassette tape recorder (software controlled), 32-column matrix printer, a full keyboard. There are three eight-bit parallel I/O ports available for general purpose use and a programmable audio tone generator and speaker within the enclosure.

The system design incorporates a full graphics system (256 x 224 resolution elements) that will run on the enclosed monitor and provides a standard full color RGB video output for external color display. The system is enclosed within a rosewood desk; the keyboard is contained within the center drawer.

Optional accessories include the full operating system and development software on PROM or mag tape, BASIC on PROM or mag tape, a PROM burner card, additional I/O ports, a second independent video display card, color monitor and RS-232 interface. Price is $3385 complete.

Noval, Inc., 8401 Aero Dr., San Diego CA 92123.

**IBM Selectric Printer Approved for Microcomputer Output**

Micro Computer Devices has announced availability of the SELECTERM, a fully converted IBM Selectric II typewriter. The conversion to a printer enables immediate use with any microcomputer.

The SELECTERM may be connected directly to either parallel or serial port, with all inputs at standard TTL level. No additional software is required since all logic is in an internal PROM. The SELECTERM includes a special typing element that produces all ASCII and full uppercase and lowercase alphanumeric characters. Also included are tab command, back space, vertical tab and bell. All necessary electronics and cable sets are provided along with documentation for unpacking, connection, testing, theory of operation and schematics. Special features may be ordered.

The SELECTERM can also be used as a typewriter. Because the SELECTERM has been approved by IBM, the typewriter warranty remains active, and yearly service contracts may be obtained from IBM. Micro Computer Devices provides a separate factory warranty on the conversion package. The SELECTERM may be purchased only through dealers, though OEM inquiries are invited. Full price is $1650.

Micro Computer Devices, 960 E. Orangethorpe, Bldg. F, Anaheim CA 92801.

**New PROM/RAM Board Kit**

Vector Graphic's PROM/RAM board occupies two independently addressable 8K blocks and has a 1K on-board RAM and capacity for up to 12K 2708-type EPROMs. Complete addressing flexibility is provided to conform to virtually any system configuration with a minimum of address jumpers required.

Video boards or disk-operating systems can be nested in the 3K of unused space; MWRITE logic and jump-on-reset allow operation without a front panel. A 24-command PROM monitor is available to interface with most popular I/O boards. $175 assembled.

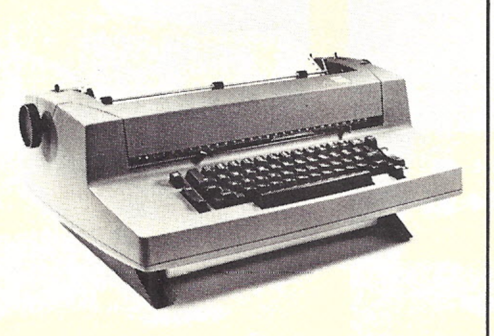
Vector Graphic, Inc., 790 Hampshire Road, Westlake Village CA 91361.

**PAIA 8700 Computer/Controller**

PAIA Electronics, Inc., has introduced their 8700 computer/controller, the OEM microprocessor development system.

Based on the popular 650x family of processors, the 8700's fully socketed, plated-through board provides spaces for 1K of RAM in 256-byte increments (2112) and 1K of PROM, also in

*Noval's updated 760 System.*



*Micro Computer Devices SELECTERM.*

five 8-bit parallel input ports; and one 8-bit parallel output port. Several connectors are provided for system expansion and the implementation of more complex I/O structures.

The PAIA Interactive Editor Debugger (PIEBUG) monitor program (256 bytes) provides complete control of code entry and debugging, and a relative address computer for automatic calculation of relative branches, a back-space key for stepping backward through memory and Pointer High and Pointer Low keys that make the 8700's twin seven-segment displays serve the multiple functions of address and data display. All monitor functions are implemented as fully documented, user-available subroutines.

Currently available options include the PS-87 power supply ($24.95) and CS-87 cassette interface ($22.50). A variety of low-cost (less than $40.00) video-display options for the 8700 is scheduled for first-quarter release.

The PAIA 8700 is available in kit or assembled form in a variety of low-cost configurations starting at less than $90.

PAIA Electronics, Inc., OEM Sales, 1020 Wilshire Blvd., Oklahoma City OK 73116.

### MTX-A1 Alpha Chip

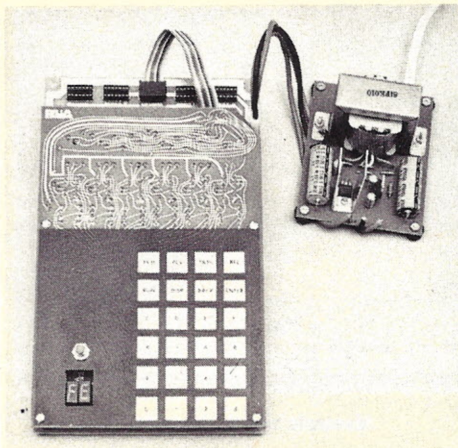
The MTX-A1 (alpha chip) is a general-purpose programmable alphanumeric display and keyboard interface device designed for use with most 8-bit microprocessors. The display portion provides all the timing and refresh signals to drive up to 32 popular 5 x 7 dot matrix LED displays. The keyboard portion provides all scanning signals to debounce and decode any keyboard of up to 64 keys.

On the input side, the alpha chip interfaces directly to the address, data and control buses of most 8-bit microprocessors. The MTX-A1 is the first alphanumeric display controller to incorporate virtually all functions on one chip.

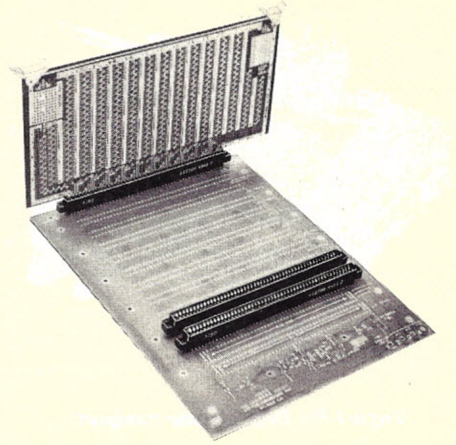
Many intelligent commands are incorporated. Examples are clear display, shift display left/right, blink cursor, read/write display, self test, etc. The display and keyboard parameters (such as the number of characters in the display, refresh frequency, the numbers of keys, etc.) are fully programmable.

The IC requires a single +5 V ± 10 percent power supply (60 mA). All display and keyboard I/O pins are TTL compatible. The MTX-A1 can be interfaced directly to any TTL, CMOS or NMOS uP through an I/O port or bus. The chip is available in a 40 pin DTP standard plastic package and has a temperature range of 0-70°C. The price is $39.

Matrox Electronic Systems, PO Box 56, Ahuntsic Stn., Montreal PQ H3L 3N5. Canada.



*8700 CPU board and PS-87 power supply.*



*Vector's model 8803.*

### S-100 Motherboard Prewired for Active or Passive Bus Termination

Priced at $29.95, the Vector Electronic Model 8803 motherboard has the features of more expensive S-100 bus motherboards but allows system fabricators greater latitude in configuration and cost. The board has positions for up to eleven 100-pin card-edge connectors. One position may be used to interconnect to other motherboards for system expansion.

Twelve tantalum capacitors are included to suppress transients on the +5, +12, and −12 volt buses. Heavy buses are supplied for ground, +5 V, and ±12 V. Ground and +5 V buses are rated at 10 A, while ±12 V buses are rated at 7 A. Tie points for power-supply sense lines permit remote monitoring for improved voltage regulation.

The motherboard complements Vector's broad line of compatible S-100-bus-oriented enclosures, receptacles, interface cards and packaging hardware.

Vector Electronic Company, 12460 Gladstone Ave., Sylmar CA 91342.

### Vector-Pak Enclosures

Two new enclosures from Vector Electronic Company give system developers a choice in packaging Imsai, Altair and other 5.31 × 10 inch (13.48 × 25.4 cm) cards with S-100-bus configurations. Vector's VP1 and VP2 cases cost $128.30 and $134.30, respectively.

They are compatible with Imsai and Altair microcomputers, the VP2 having front-to-back card orientation and the VP1 having side-to-side card orientation. An aluminum chassis at the rear or side supports power supplies and other heavy components.

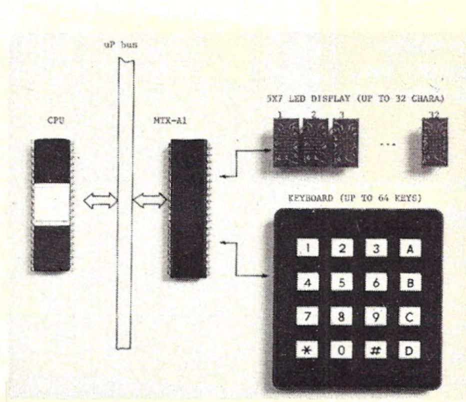
Two dozen plastic guides are supplied so the user may position 12 cards in any location and with any board spacing. For expansion, the cases have space for 21 cards on 0.75 inch (1.90 cm) centers. Adjustable slots allow convenient mounting of receptacles or a motherboard. Both cases are 17.85 by 9.01 by 17.1 inches overall, and weight is 15 pounds.

Optional accessories include a pre-punched rear panel with ten holes for 25-pin D-type connectors, Vector's 8800 series prototyping cards for S-100-bus systems and a wide variety of sockets, connectors, pins and tools.

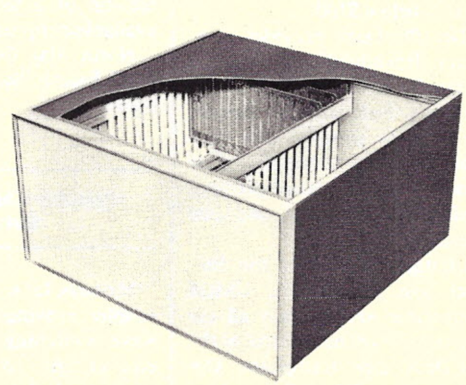
Vector Electronic Company, 12460 Gladstone Avenue, Sylmar CA 91342.

### COMPTRONICS F-8 Development Board

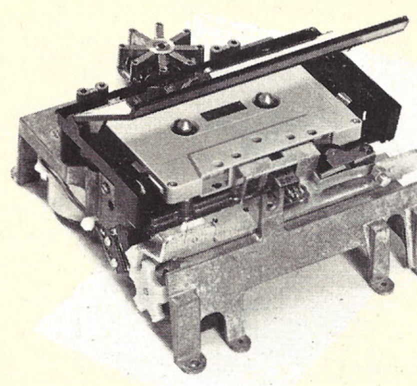
COMPTRONICS has put an F-8 microcomputer on a single board. Designed especially for low-cost hardware and software development and evaluation, the Model 1080 F-8 Development Board consists of an F-8 CPU, a FAIRBUG PSU, a 3853 SMI, 2K x 8 of RAM, 2.0 MHz crystal and



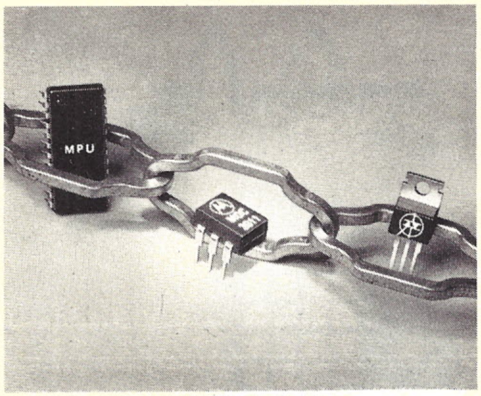

*Matrox Alpha Chip.*



*Vector VP2 enclosure.*



*Triple I Phi-Deck cassette transport.*




*Motorola's optical coupler.*

interfacing componentry on an 8 x 13 inch printed circuit board.

Aimed primarily at the design engineer, experimenter and serious hobbyist, the development board also contains a buffered address and data bus to an S-100 memory expansion connector, and provides sockets for 4K of 2708 memory. The unit provides 1K of 2708 user custom monitor, and has 32 bits of I/O arranged in four 8-bit ports.

The microcomputer provides for RS-232 or 20 mA current loop support circuitry, two sockets for I/O expansion and many other features. Complete documentation is included in the basic price: $249 kit; $299 assembled.

COMPTRONICS, 19824 Ventura Blvd., Woodland Hills CA 91364.

## New Phi-Deck Cassette Transport

Triple I, Inc., is adding a new fixed-speed model with an ac capstan motor to its line of electronic cassette tape transports. Features of this model include four-motor control, remote-control capabilities, fast start/stop, less than 30 seconds rewind, and speeds from 1 to 10 ips. The price for a single unit is $149. Quantity pricing for 500 units is below $100.

The Phi-Deck provides maximum dependability because its design incorporates fewer moving parts. Four separate motors, which allow complex tape-deck functions to be accomplished by remote control, control take-up, rewind, play or record, and head engagement.

Control boards for the Phi-Deck are TTL, DTL, CMOS compatible and contain all the circuitry for proper control of the Phi-Deck tape transports. Options such as BOT/EOT sensing, record/play, read/write electronics, cassette-in-place sensing and others are available.

Triple I, Inc., 4605 N. Stiles, PO Box 18209, Oklahoma City OK 73118.

## North Star Horizon Mainframe

North Star Computers, Inc., manufacturer of microcomputer peripherals, is entering the microcomputer mainframe market. The new North Star Horizon computer uses a full-speed (4 MHz) Z-80 microprocessor and includes 16K bytes of memory, a disk controller with one or two Shugart minifloppy disk drives and full extended disk BASIC. A serial I/O port is included for connection to any standard baud-rate terminal.

Options for the Horizon include additional disk drives, hardware floating-point arithmetic board, 24-line by 80-character uppercase and lowercase video display controller (VDC) board and 16K memory board with parity check. The VDC board, when used in conjunction with a North Star 16K memory board, will display high-resolution (480 by 250) graphics on a TV monitor. The computer uses the widely supported S-100 bus, allowing possible use of a large selection of available peripheral products.

North Star Computers, Inc., 2465 Fourth St., Berkeley CA 94710.

## Optically Coupled Triac Driver

Motorola's new optical coupler provides 115 V ac full wave switching and isolation equivalent to an electromechanical relay at the command of a low-level dc source such as IC logic. Used alone, the MOC3011 switches power-line loads up to 7½ Watts. Kilowatt loads are switched with a power Triac, directly driven by the MOC3011. Bidirectional Triac-like output characteristics of the coupler eliminate the complex interface circuitry previously required for photocouplers having unidirectional transistor or SCR outputs.

The 6-pin DIP MOC3011 encloses a gallium arsenide LED which is energized by input currents of 10 mA at voltages as low as 2 V. Photons emitted by the LED travel through a clear insulator capable of withstanding 7500 V. This triggers a unique monolithic photosensitive chip in the same DIP, whose output simulates a small bidirectional Triac capable of switching power triac input or small load currents up to 100 mA and sustaining output voltages up to 250 V in the "off" condition.

The MOC3011 makes practical, direct control of 115 V ac alarm lamps, transducers and small appliances possible by microprocessors or remote low-voltage switches. By adding a power Triac, the same sources can switch motors, heaters, solenoids and other heavy ac loads. Price is $1.60.

Motorola Semiconductor Products, Inc., PO Box 20912, Phoenix AZ 85036.

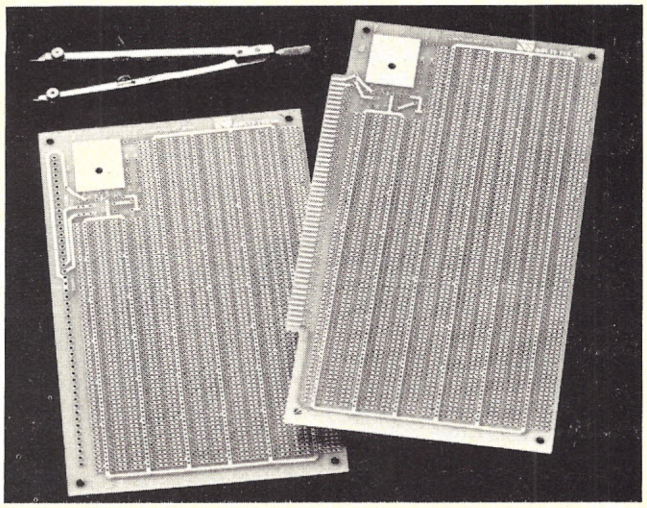
## Multi-Tek Prototype Boards

Multi-Tek, Inc., offers serious computer and electronic hobbyists and professionals four new prototype boards. Before Boards, with plated-through holes, are built to the highest government standards. They are ideal for analog or digital circuits and can be either hand-wired or wire-wrapped. All boards are designed to accept flexible cable connectors for piggyback applications.

The S-100 Before Board ($29.95) is supplied with a 50-pin gold-plated edge connector and can accommodate 56 (14-pin) or 50 (16-pin) IC packages. It has a built-in regulated power-supply circuit and is compatible with all S-100 bus systems. The S-100x Before Board ($28.45) is identical to the S-100 but does not have the 50-pin gold-plated connector or a regulated power-supply circuit. The S-100x thus lends itself to laboratory and testing applications where the quality of the breadboarding medium is of paramount importance.

The 6800 Before Board ($23.95) is compatible with all 6800 bus systems, and is designed for use with 50-pin Molex connectors. The 6800x Before Board ($22.46) has a larger hardware-carrying capability and is not furnished with the regulated power supply circuit.

Multi-Tek, Inc., PO Box 201, Union Square, Milford NH 03055.



*Multi-Tek Before Boards.*

# LETTERS

### Arbitrary, Subjective and Misleading?

I have just read the December 1977 issue of *Kilobaud*. I must take exception to "Compleat Guide to Logic Diagrams" by Russell Lauffer. Several of the ideas put forth in that article are arbitrary and subjective. As a result, they tend to be misleading.

The facts are that "mismatches," as they are called in the article, are *not* incorrect drawing procedure. As a matter of fact, most logic-design engineers pay no attention to such trivia as long as the circuit is logically correct.

Also, a circuit containing all NAND gates does *not* imply that the person who drew the schematic was saying all of the logic functions performed are AND. It probably means that the designer was able to complete the design with a minimum number of chips using NAND gates and proceeded to represent the gates just as they are shown in the vendors' catalogues.

Once a person is familiar with standard logic symbols and their functions, he may then translate NANDs to ORs, etc., if he feels it helps him interpret a schematic more readily. However, most people do not feel this inclination and, as a result, continue to use standard logic symbols as given in manufacturers' catalogues.

To suggest that one way of drawing a gate is wrong and another is right is untrue. And, it is unfair to those who are just beginning to understand logic diagrams.

In general, the best rule to follow is to draw a logic diagram in as clear, direct and simple a way as possible. The left-to-right flow rule is the most widely accepted rule in drawing any schematic. Most other rules tend to be arbitrary and constraining. There is no clear-cut right or wrong way. To suggest there is is misleading.

**Robert H. Penoyer**
**Monterey Park CA**

*I may disagree with what you say, but I will defend . . . John.*

### Corrections

Since the publication of my KIM expansion article in the January issue of *Kilobaud*, I have found a manufacturer who will etch and drill a PC board for the article. The prices are: Undrilled board \$3.80; Drilled board \$6.50; Parts kit (no board) \$22.50. The address is: O. C. Stafford Electronics Service and Development Co., 427 South Benbow Rd., Greensboro NC 27401.

I hope this service will allow KIM-1 users who are not set up to etch their own boards to build the project.

**John Eaton**
**1126 N 2nd**
**Vincennes IN 47591**

### AppleSoft Benchmarks: Fast!

I was disappointed when I read "BASIC Timing Comparisons" in the October issue. The authors decided not to include the integer-only BASICs, possibly because of comparing apples with oranges. Not so—if they're properly documented.

I own an Apple-II, allowing me to choose either an integer-only or floating-point as the need arises. Except for scientific programs, the integer form is more than adequate and very convenient. But let's not kid ourselves—most home and small-business systems are used for games, inventory and money management, and control of external devices, which can be handled by an integer BASIC in most cases.

For those readers who would like the benchmark timings of the fastest integer and fastest floating-point BASIC (at 1 MHz), I have provided the figures in the accompanying table.

Yes, the timings and the name of the floating-point indicate this is the latest version of Microsoft for the 6502. This system runs with a 1 MHz clock and uses the *500* ns memory chips while using a memory-mapping technique and *no* hidden refresh cycles!

I really enjoy *Kilobaud*, but I wish your authors would not delete any highly competitive systems from their articles. After all, an eight-pound apple carries more weight than a three-pound lemon.

**Tom Scogin**
**Atlanta GA**

*Apple's new floating-point BASIC from Microsoft is a fairly new development, and I suspect Tom Rugg and Phil Feldman simply hadn't heard of it when they prepared the update in issue No. 10. They didn't deliberately leave any companies out. The Apple-II is a* fine *system . . . as a matter of fact, I'd like to see more articles describing what people are doing with it.—John.*

| Benchmark Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Apple-II integer | 1.3 | 3.0 | 7.2 | 7.0 | 8.8 | 18.4 | 27.8 |
| Apple-II AppleSoft | 1.3 | 8.7 | 16.0 | 17.8 | 19.1 | 28.3 | 44.5 |

### Reviews . . . and More Reviews

I'm just getting into computers, though I've been active in audio electronics for nearly a decade (yes, I know about *The Audio Amateur*—I love it!). *Kilobaud* has been a helpful addition to my magazine library, and I'm going to start a subscription for my brother-in-law, a NASA engineer who's getting heavily into solar-energy applications.

Because my income is limited, any hope I have of getting a system together rests in buying bits and pieces at a time. One of my first purchases will be an S-100 bus, since that seems to be a convenient "standard" of some sort, and lots of people supply all sorts of cards for it. I bet I'm not alone in this.

I notice that several sources exist for the S-100, along with the usual varying claims for superiority. Godbout looks like one good source and Thinker Toys has what looks like a really good layout, with interleaved grounds for improved shielding. Active terminations look like another plus, too. But I don't have any real way of comparing them all to determine which one(s) is/are superior. I'll have to do a lot of reading and research to make a decision. Comparative articles on this, CPUs, memories (Mark Garetz's article in *Kilobaud* No. 13 is a good example), I/O, power supplies, front-panel displays, TVTs, keyboards and gosh-knows-what-else would be a big help. There is, after all, some controversy as to whether or not "S-100 compatible" really is compatible. Personally, I want to go with a Z-80 uP. But I might change my mind later.

For those readers visiting the Orlando FL area, a visit to Skycraft might be worthwhile. It's the best surplus house I've seen yet. You can leave the wife and kids at the Mouse Factory and run back up I-4 and spend an afternoon having *your* kind of fun.

**Damon Hill**
**Atlanta GA**

*Keep reading, Damon. The articles and reviews you're looking for are on the way!—John.*

### Not Problems—Future Improvements

I am writing in response to your editorial request to hear from Radio Shack TRS-80 owners. To put my remarks in perspective, you should know that I have been involved in DP for 15 years, mostly in software and management, and have no previous microcomputer experience except reading.

I've had my TRS-80 since about September 20th, and am very satisfied with it. Delivery was prompt (about six weeks), and the local Radio Shack people, though not very knowledgeable, have done everything possible to keep me happy with it. The TRS-80 has been *extremely reliable*; to the best of my knowledge, I have not had a problem that couldn't be traced back to operator error. I am surprised at the speed and capabilities of the 4K BASIC (including *data* read and write to cassette, PRINT AT, ON X GOSUB, etc.), but don't have room to detail all the good points. I will, however, for your readers' interest, list what I feel are the TRS-80's four major shortcom-

ings and a few of the trivial ones.

1. *Only one tape unit*—can't do true data file processing. My solution: *Add second tape for about $12* and cost of recorder.
2. *Can't get to Z-80*—no PEEK, POKE or CALL commands; no assembler (as of November 1977). Solution: According to what Radio Shack said then, the assembler would be available by the end of 1977.
3. *Poor alphanumeric capabilities*—only two string variables, no alpha subscripting, no compare, etc. Solution: assembler or Level II BASIC.
4. *Documentation*—preliminary manual barely adequate to use BASIC; final version (I've seen draft) covers only BASIC—still nothing on hardware, Z-80, monitor, bus, etc. Solution: Ask for?
5. *Trivial*—single-letter variable names, a single one-dimensional data matrix, slow tape I/O (300 baud), 6-digit precision, limited number of built-in functions, graphics resolution only 48 x 128. Solution: For only $600, I can live with or work around these. I have not seen a BASIC program I couldn't convert for the TRS-80 (if no special input/output devices are required).

One other thought has occurred to me. I am amazed at the size of a program that will fit in the standard 4K memory! The ability to abbreviate almost any command is very useful. Unless your application requires large amounts of data in memory at one time, it seems that you could program the world in 4K! (Remember the old 1401s and 650s?)

I am currently working on the *structured design* and programming of a personal-finance package and a full-blown (64-quadrant, 64-sector, graphic) Star Trek.

I enjoy your magazine very much. For me, it has the best mix of software and hardware articles. Keep up the good work.

**Jim Green**
**Lawrence KS**

*Just wait, Jim. Level II BASIC will bring much happiness into your household!—John.*

### "Do I Need A 360?"

I find *Kilobaud* interesting as I used to find *QST* interesting—having articles I might understand some day. Perhaps someone could write an article really explaining what one or maybe several different microcomputers can really do—I mean, for the appliance operator. I like the reading material, all about the hardware used in these gadgets, but I don't intend to buy a micro to fool around with the hardware; it's cheaper to build digital clocks and such things.

The answer given to "What can they do?" is usually "Your imagination is the limit." However, for me, and perhaps for most of us, that isn't true. The true answer is: "Your money's the limit." I'm afraid it may take an IBM 360, or such a machine, to do as I'd like (I haven't found the answer to that yet).

The article by Sherman Wantz on using the microcomputer for a memory (issue 11) was the best in the way of giving an answer. We all know we can play games with the things (a checkerboard is cheaper), but now we know it's possible to get lots of things into digital form on a cassette cartridge. The article wasn't too clear about getting it back out, or how much one can even get in, but I presume it's applicable at least.

How far can this one (useful) aspect of microcomputers be stretched before you need a biggie? Could I put all my files on tape or disks (and still have room for the computer in my shack)? How about a dozen or so books on tape? How about indexing and cross-indexing all of these—here's where the article got sort of hazy and noncommittal. Is this possible with a microcomputer, a floppy disk and (to get the stuff back out) a CRT display and maybe a printer? Or do I need an IBM?

I think what I'm asking is—how limited are these things? Does what the manufacturer put into the CPU determine how much of this you can do? Or can you get a microcomputer that can be programmed by any language you want or need, to do what you want it to do? Is this why the Ohio Scientific Challenger III has three CPU boards?

I realize you don't answer letters except by letting them be read—but neither do the articles you print answer them. Do we have to learn by little bits at a time, one bit from each article, perhaps, till we get it all together? Maybe in the far future, Kilobaud Klassroom might get around to some answers to such software problems. Meanwhile, I'll have to wait before I get one of these gadgets—maybe by the time I find out they'll work for my needs (or won't work), they'll be cheap enough that I can even afford one. That'll be the day!

**Michael Windolph**
**Chaska MN**

*Well, Mike, I have one answer for all those questions. You have to get a computer. The fun won't begin, nor the mystery be dispelled, until you start pounding a program in on a keyboard (and get all those error messages). Then you'll start finding out just what your processor can, and can't, do . . . how much memory you're going to need . . . what kind of mass storage will suit you best . . . and what kind of terminal and printer you'll require.*

*I used to teach computer courses. The last thing in the world I would ever attempt would be to teach a computer course without a computer. I have similar feelings about a person trying to simply read about computers . . . and gain an understanding of them. It's not that simple. You need to go beg, borrow, steal, or buy a system . . . but get one! (But don't get an IBM, OK?)—John.*

### The Computer Hobbyist Lives On!

The Computer Hobbyist's reported death (Frederick Holmes' letter, *Kilobaud* No. 13) is inaccurate. TCH is alive and multiplying, and has moved to San Luis Rey.

Subscribers to the original TCH are receiving the "Classified Advertiser" edition as subscription fulfillment. This monthly publication brings low-cost advertising to hobby computing. For $2, anybody (subscriber or not, commercial or hobby) gets a five-line classified, with guaranteed 3000-minimum paid circulation. Being *classified* (with headlines and organization), it is easy to read. Subscriptions are $5. If an address change is requested during the first year of subscription, expiration is one year. If no address change has been effected after a year, service continues indefinitely, until address change.

*Word Processing Letter*, $12.95 for 12 issues, will move under the new nameplate, as has *2650 Computer User Notes,* which will be joined by two newcomers, *S-100-Bus User Notes* and *S-50-Bus Computer User Notes*, all of which are $5 for six issues.

The user-note publications, which are free of advertising, are able to report problems with products and vendors, as well as solutions to these problems. The fact that TCH has an in-house advertising publication does not interfere with this objectivity, since the advertiser is based on a heavy volume of $2 classifieds. (We aren't likely to be swayed by the threatened loss of a $2 ad.)

Our policy on complaints is to notify the vendor, and allow two weeks for reply. After that, we feel free to publish whatever we have on the subject. These publications will be of considerable service to the hobbyist.

**Bill McLaughlin, editor**
**The Computer Hobbyist**
**Box 158**
**San Luis Rey CA 92068**

### Business Applications

I was glad to see in the January issue of *Kilobaud* your article for business applications, and I certainly look forward to articles written by users of such systems. Unfortunately, there is not as much incentive for users to write articles about the systems they have developed as there is for developers promoting their own products. I certainly hope you find some. In the event that users who have developed their own systems do not come forth with articles, perhaps you could take the approach of using articles by other vendors, so that, at least, we could be informed about alternatives, should we decide to take that route.

Personally, we are committed to developing our own system, and to date are about 90 percent complete. I am looking forward to more articles in this area.

**Seaton T. Preston**
**President**
**PolyScience Corp.**
**Niles IL 60648**

*I would prefer to see objective reviews in the business-applications section . . . and that's what I'll be going after.—John.*

### Autopilot for Lunar Lander?

I enjoyed Mark Borgerson's two articles related to a real-time lunar lander. The first article (*Kilobaud* No. 3) has an error in the schematic. CB1 should be CB2. The second article (*Kilobaud* No. 12) has an error in line 190. The element V+SQR should read V-SQR.

A fun trick in the latter article

is to let G = 0 (gravity). If you don't work it just right, you *never* get down! Also, the burn input factor can be changed in line 110 to give more thrust (2.4 is used). The H (height) and V (velocity) can be changed to random integers so a lander can't utilize the fact that 26.8 = B (burn) will always land you just right every time.

No doubt some spoilsport will already have discovered an autopilot to override the joystick input for a perfect landing for any input H, V, etc.: #112 LET B = (W*((2*G*H) + (V*V)))/(2400*H). The mentality of a person who would develop an autopilot for a fun game like this is the same type of mentality that would be so annoyed by a perfect QUBIC program (3-D ticktacktoe) that he would set his computer up against a friend's also programmed with QUBIC and watch them go at it. (Cat game, obviously.)

**David O'Neil**
**Greenacres FL**

### 3-D Graphics Fun

I would like to acknowledge the quality of the three-dimensional-graphics package offered by Sublogic described in a recent *Kilobaud.* I had little difficulty implementing the package since the documentation was quite clear and very complete. I am now using it for a primitive flight simulator and I have begun designing a STAR WARS game.

**Alan Freiden**
**Reston VA**

*Well, don't just sit there programming, Alan! Write an article about that flight simulator and game, and share it with the rest of us!—John.*

### An Insult to the Programming Profession

In reference to the Editor's Remarks on pages 6 and 27 of the December 1977 issue of *Kilobaud,* I think your suggestion for "Low Cost Software Development" is ill-advised and is an insult to the computer-programming profession.

There is much more to a well-designed software system than just sitting down and coding a program. First, the programmer-analyst must have a knowledge of the application for which a software system is to be designed. How can a person program a general ledger system if he or she knows nothing about accounting? How can that person design an operating system without knowledge of what an operating system is supposed to accomplish? Many high-school and college students are very intelligent, but the "crash" courses in computer programming usually offered these days do not teach more than how to write program code for mathematical problem solving and game playing.

Software intended for customer distribution must do more than simply allow a computer to perform some function. First, it must perform to some specification, and that specification must be properly designed and documented. In other words, computer programs must *work,* and work as specified. How many complaints have magazine editors received about manufacturers' software that didn't work as the user thought it should? Debugging software—especially systems software such as disk I/O routines, interpreters and compilers—requires an expertise born of experience and proper training, just as does the debugging of hardware.

Second, computer software should be properly documented. It is not enough that the author understand it. Not all people think the same way, and not all people are equally intelligent. The same principle applies to hardware and software. How many complaints have the same editors received regarding poor documentation? I see them in almost every hobbyist computer magazine I pick up. Do high schools teach documentation? Do they teach "human engineering"? If you don't believe human engineering is important, consider the plight of a beginning programmer typing in a command to a very popular development system and getting the single response WHAT?

I have nothing against high-school students—or anyone else—learning to program. But, there is an old saying: "You get only what you pay for." If hardware manufacturers continue to view computer software as garbage, to be provided only to the minimum extent required to pacify a few customers, then that is what they will get—garbage. It is time the manufacturers begin to place at least equal importance on their software as they do on their hardware.

I have 12 years' experience in the data-processing industry, working with both high-level and low-level languages in manufacturing, accounting and systems applications. If you were a manufacturer would you rather get software from an experienced professional or a hobbyist with six months' experience?

Now, let me ask you another question. If you just had a heart attack, who would you rather have operate on you—a well-known and established doctor with 12 years' experience, or an intern?

**Charles Pack**
**Los Altos Hills CA**

*Send the closest one . . . and the one who charges the lowest rates! And, when it comes to sending systems to high schools with the possibility of getting some good software developed (and providing the students with systems from which to gain knowledge and experience), I still say it's a good idea. It is possible that my local high school is an exception . . . but I doubt it.—John.*

### Comments on a Minimal Usable System

I have been prompted to write by your request for definitions of minimal "usable" microcomputer systems in the Publisher's Remarks section of the January issue (No. 13). First, I think I should explain where I am coming from, to put my remarks in some kind of context. I am a systems programmer (of the operating systems variety, not application systems). As such, I primarily use micro systems for development work, not "end-user" applications. However, I feel that I do have a pretty good view of what should comprise a usable system.

With the present state of the art, I really don't think cassette tapes make a good primary storage medium from several standpoints: (1) they just aren't that reliable (correct me if I'm wrong); (2) they are slow; (3) they are awkward to use (from the standpoint of the "computer-naive" average end-user). For these and other reasons, I think that for business systems (and frankly, business systems are where the largest volume of computation gets done), disks, floppy or hard, are the only feasible program and storage methods. I may be biased toward disks since where I work, we have four micro systems (all 8080/Z-80), and all of them have two floppies. Speaking of which, I also think that two drives are almost a must.

So much for mass storage, now to the nitty-gritty. Memory. I think that for many significant operations, at least 32K can suffice. Most applications wil probably fit in 48K, but it is amazing how fast memory can be eaten up. Two of the above mentioned dual-floppy systems have 62K RAM, and I have a BASIC program that would require at least 140K to run. And that is "only" a game (Star Trek). Needless to say, I gave up on it a long time ago (who wants to convert 5000+ lines of BASIC into assembler for a game).

As far as I/O equipment goes, I think that most business applications require hard copy, yet should have a CRT console. One configuration that we are very happy with is a Teletype Model 40 line printer (300 lpm) and the omnipresent ADM-3a. The Model 40 only costs about $2800, ready to plug in and go (from TTY, which unluckily has a three to four month lag time on shipments) and is probably one of the solidest printers I have ever used (and I spent several years in IBM shops). For low-volume hard copy, there isn't much choice except matrix printers at a low cost.

As for mainframes, I will just make the following comments. The S-100 bus has many apparent advantages (mainly low cost and wide variety of boards available) but has the BIG problem of reliability. It takes practically an electrical engineer to make a system that looks nice on paper work. I think I can recommend the Intel MDS-800 without reservations, but it costs a not-so-small fortune. We have an Imsai, which works very nicely, but we initially had trouble with the disk controller (no names). I have a friend who has a Vector Graphic box, CPU and 64K RAM, and a Digital Systems dual-floppy. It worked from the time he plugged it in and turned it on, so I guess it's a case of getting what you pay for.

I think the biggest problem the businessperson has is applications software. Basically, there isn't much good stuff, and what there is costs $$. This, however, is improving.

At least there is good systems-software available today. For instance, there is at least one excellent FORTRAN compiler on the market, many good BASICs (interesting how many of them are Microsoft's?) and at least one good DOS. And I think that this is improving daily. I have heard that an APL is coming out, as

well as COBOL, PASCAL and other languages. So things don't look so bad after all. After all, it takes good systems and compiler/interpreters to implement application systems. Few end-users would write an accounts receivable in hex machine language.

To sum all this random noise up, and to get back on the track, it is my opinion that the following would make a good system.

1. An S-100 mainframe (pick one, any one).
2. 48-64K RAM from the same manufacturer as the CPU.
3. A conventional CRT (such as the ADM-3a, SOROC, etc.).
4. Dual-floppy drives (preferably IBM format).
5. Some kind of hard-copy device, preferably an impact printer, so that no special paper is necessary, and preferably of the non-matrix variety for legibility's sake.

All this would probably cost anywhere from $6000 to $10000 (no small chunk for a hobbyist, but reasonable for a business). I do think that you probably are better off in the long run to stick with the same manufacturer for as much equipment as possible.

I hope some of all that I said was of value.

**John R. Pierce**
**Digital Research**
**Pacific Grove CA**

PS. Any opinions expressed are my own, and in no way reflect on Digital Research.

## KB Content Problems (?) and Solutions

You have a fantastic magazine; however, there are some problems for the beginning computer hobbyist that your magazine and others like yours are compounding. Presented here are a few of the problems and some simple solutions.

1. *Construction of the small computer.* I (and a large group of potential computer hobbyists) do not have the large sums of capital necessary to purchase a complete computer kit with necessary peripherals at one time. However, we do have the small sums available to purchase parts as needed for our digital creation; and most of us are not going to open a charge-a-kit account for $1200 with a well-known company to get roughly $500 worth of parts and a pretty case. Therefore, it will be to your advantage to publish construction articles that give all details (schematics, PC board layouts, parts list, detailed operating instructions) concerning the construction of the small computer and necessary peripherals keeping the cost as low as possible.

2. *Programming instructions.* The Kilobaud Klassroom is great for those just starting to dabble with electronics as a hobby. Do the same for programming: Run a series of articles teaching people to write their own programs. Also, when you publish a program, go step by step and explain exactly what is happening inside the computer.

3. *Wasted space.* Stop wasting pages showing how to construct someone else's kit. Instead, let the company send the prospective buyer all the information he or she will need. That is why companies have customer service departments.

This letter is intended as a reply to "How Can We Get More Hobbyists," issue No. 9 (September 1977), page 2. In the near future I will attempt to write a series of articles that will entail 1 and 2 above; and I hope other hobbyists more advanced than I will do the same.

**Walter Hynson**
**Magnolia DE**

*I don't like "then-I-installed-all-the-ICs" reviews any more than anyone else, Walter. And, who cares if a couple of capacitors or resistors were missing? I may have let a few like that slip by . . . but I hope not.—John.*

## A Bridge to Byte?

It has been my intention for a long time to write you and let you know my opinion of your fine magazine. I am a computer programmer with 13 years of experience. Nearly all this experience is on large machines. Ever since I started in this field, it has been my dream to have a computer in my home. It is nice to see this day arriving. But I am not much in the field of electronics, hobby or otherwise. I have never put a kit together, nor would I know which end of a soldering iron is which.

My computer hobby magazines include *Kilobaud, Byte* and *Creative Computing. Kilobaud* best meets my needs as being something I can usually understand and carrying articles that are of genuine value to me, although I intend to keep subscribing to all three for the foreseeable future. I have sampled single issues of *Interface Age, ROM, Personal Computing* and *Popular Computing,* but they don't seem to be what I am looking for. Also, there is a limit to how much can be read in one month.

*Kilobaud* began very well, but I think I have detected some drop in quality of the articles as the months have passed. Lately, there have been too many articles that have been too short, too trite or too "haven't I read this somewhere before?" in my judgement. But it is clear that the blame for this lies with readers who are capable of writing but do not. You can't produce a quality magazine if all the writing must be done by a small cadre of writers while the rest of us just subscribe and yell, "MORE, MORE!" In my case, I hope to start rectifying this situation as soon as I get my own computer, probably a PET, in the near future. I have written a tutorial article on hobby computers that won second place in the computer-paper competition at the agency where I work, but it was too simple for a real hobby magazine; I could not bear the thought of submitting it and having my name associated with the same old story told a bit differently. When I have the real thing, I'll get in touch with you.

I passed over the Kilobaud Klassroom series with hardly a glance. Did not think it could do me any good. Now I am not so sure. It is a good thing that I have all the issues of *Kilobaud.* Perhaps someday my knowledge of hobby electronics will be sufficient for me to do useful things with it.

I'm sorry to hear and note all the infighting between the hobby computer magazines. It looks to me as if the field is glutted with them and not all can survive. So I reckon some people are helping the law of survival of the fittest along. My three-year renewal to *Kilobaud* must give you my opinion of its chances. But I wish I knew for sure which ones will last so I would dare to buy lifetime subscriptions to them.

Is it possible that your biggest difficulty is to avoid being a bridge to *Byte*? That is my suspicion. You may increase the competency of droves of people like me to the point that most of *Byte* becomes understandable. They may then ditch you for *Byte* without even saying thanks. On the more optimistic side, the survival of *Kilobaud* may be assured as long as it's the only magazine that carries the Tri-Tek ad.

**Robert Rockwell**
**Glen Burnie MD**

*The discord between some of the computer hobbyist magazines is something I have no use for. It doesn't contribute a thing toward furthering the goal most of us share: getting as many people as possible involved in using and enjoying personal computers. No, Bob, I'm not concerned with* Kilobaud *being a bridge to* Byte.*—John.*

## Math Rears Its Head Again!

I recently received a "How to Write for *Kilobaud*" and found your philosophy rather interesting and somewhat puzzling. Your statement "minimize math" surprised, as well as disgruntled, me. That math "scares readers" and that "they don't want to" use math seems very perplexing. How in the world can you talk about computers and computer programming without using math? Doing math, it seems to me, is what computers are all about. Aren't they? Maybe I'm wrong. But as far as this one reader is concerned—let the math roll! The more the better. Some things cannot be adequately explained without math; and such things, I hope, will be covered by *Kilobaud* in the future. So bring on the math; it is really not that horrifying.

**Rob Cave**
**Irving TX**

*No problem, Bob. You, or anybody else, can write an article for* Kilobaud *describing a practical application for personal computers that uses* any degree of math *(high or low level). Just make sure you don't forget that practical application, OK? We don't really need math just for math's sake, do we?—John.*

### *Letter Submissions*

*Please type (use capitals and lowercase, and double space) letters addressed to* Kilobaud *and intended for publication. We appreciate and depend on reader feedback; and use of a caps-and-lowercase, double-spaced format by readers will immeasurably facilitate the editing and typesetting process. Thank you, readers-writers.*

# Build the "Simple Computer"

## *a home-brew 8080*

*Dick Whipple*
*305 Clemson*
*Tyler TX 75703*

Just how difficult is it to build an Intel 8080 microcomputer? That was the question rolling around my brain a few months ago. At the time, I was laboring under the impression that such a task would involve considerable engineering and technical skill. I suppose the project would have ended right there had I not been in the midst of a slight financial crisis at the time. I was ready to have my own microcomputer, but there simply wasn't enough in the kitty to buy an Altair or Imsai. Could I put together an 8080 system that would function like the commercial ones, but cost half as much?

Well, today I am pleased to report to *Kilobaud* readers that it can be done! With an investment of time and about $600, I put together a system that is functionally equivalent to a $1300 Altair system belonging to a friend of mine. What did I get for $600? I'll tell you, starting with my configuration.

- 8080A CPU with EPROM for start-up (no front panel)
- 17K static memory
- 32-character by 16-line TV interface
- 1100-baud cassette interface
- ASCII keyboard
- Baudot Teletype for hard copy

Although digging into the junk box helped keep the cost down, the real money saving came from simplifying system design. In the article that follows I want to emphasize the design areas that reduce the complexity of an 8080 based microcomputer. Although I will give some construction details, the article is not intended as a complete construction guide. However, with the ideas presented and a little technical skill, you could build the system and save some money too!


*Another use for home computer. Many high-school math textbooks include BASIC programs. This course in trigonometry uses several programs to introduce students to the use of computers on their own problems.*

### System Description

The first stage of the design was a CPU board. Since the 8080A microprocessor was my choice, I picked up a copy of the *Intel User's Manual.* Leafing through it, I made an interesting discovery—the 8228 system controller chip. No one had told me about this little jewel, but there it was—bigger than life. Reading further, I made a second important discovery: Microcomputers don't have to be complicated as long as you take the right approach to them. What is this approach, you might ask?

To begin with, you must establish in your mind that the microprocessor has four fundamental links to the outside world. As shown in Fig. 1, two of the links provide a data path to and from memory, while the

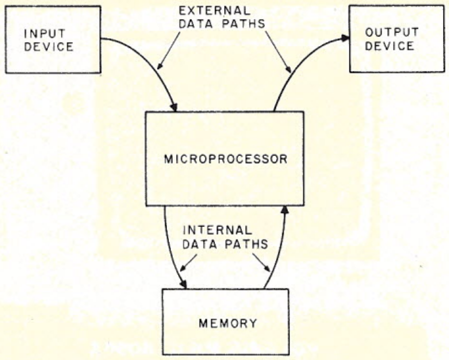



*Fig. 1. A simplified view of the microprocessor and its relationship to the outside world.*

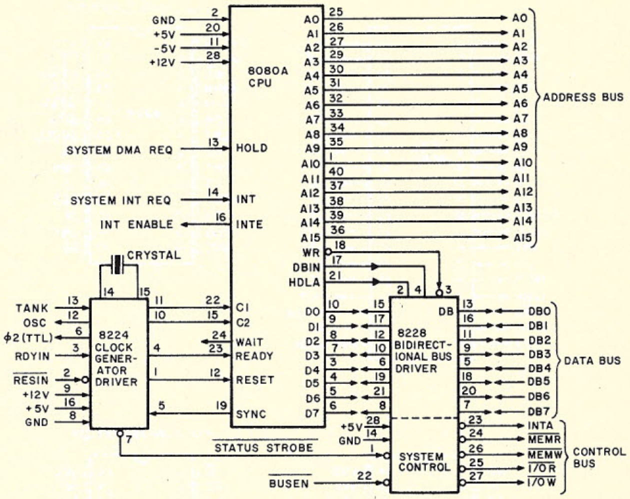



*Fig. 2. 8080A CPU using the 8228 system controller and 8224 clock chip. (Reprinted from the* Intel 8080 User's Manual.*)*

other two are paths to and from I/O devices. What data passes over these paths depends on the programmer and his software, but that's not our concern at this point. Based on this "bare-bones" structure, the CPU board must be capable of providing: (1) a 16-bit memory address and an 8-bit I/O channel address; (2) an 8-bit data bus—with data flow both in and out of the microprocessor; (3) signals to control (1) and (2) above, namely—Memory Read, Memory Write, I/O Read, I/O Write. Of course, there are lots of other signals that you can create for this or that use, but when you get right down to it, most are not essential to basic microcomputer operation.

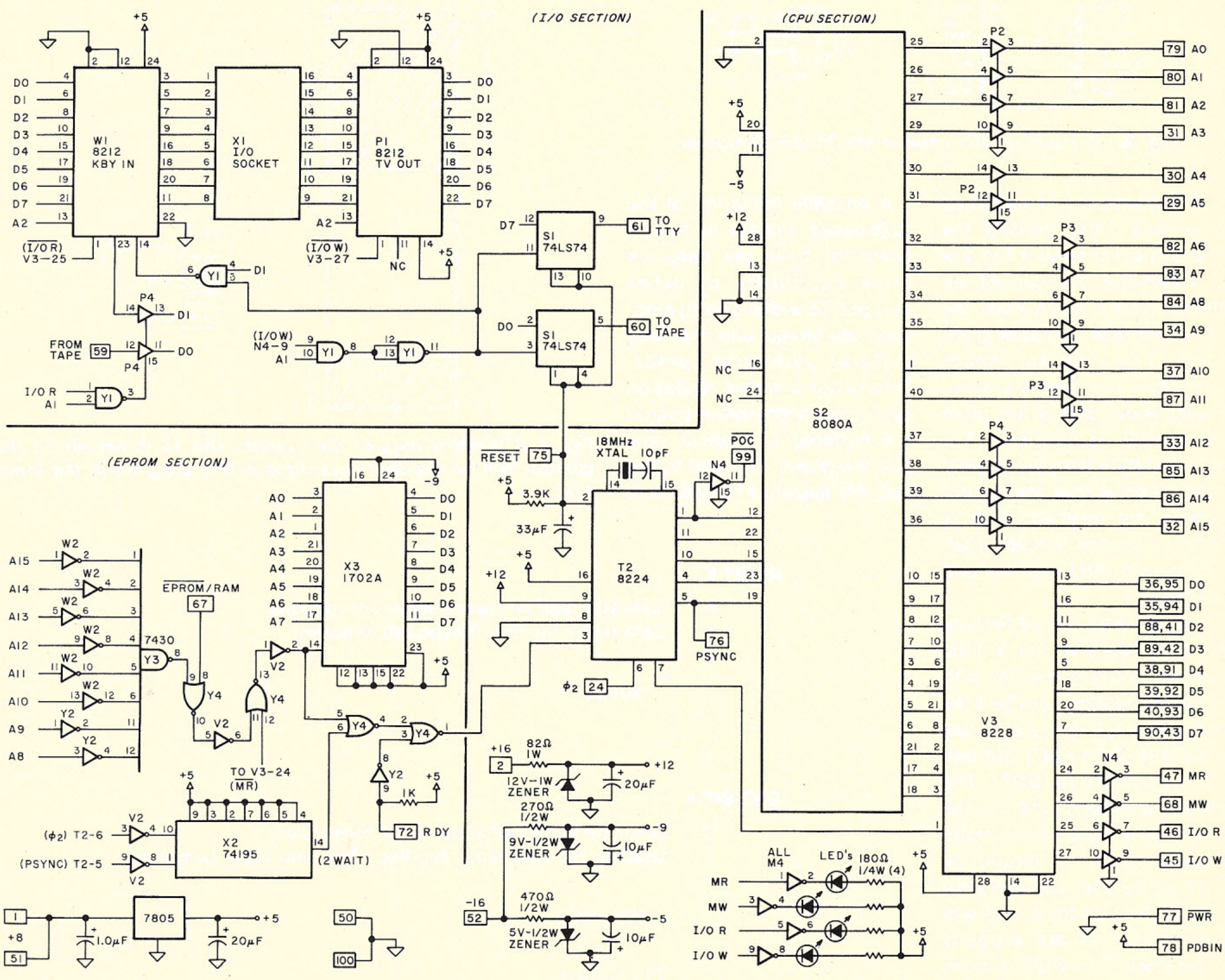



*Fig. 3a. Complete CPU with EPROM and I/O circuits. IC location coordinates refer to Fig. 6.*

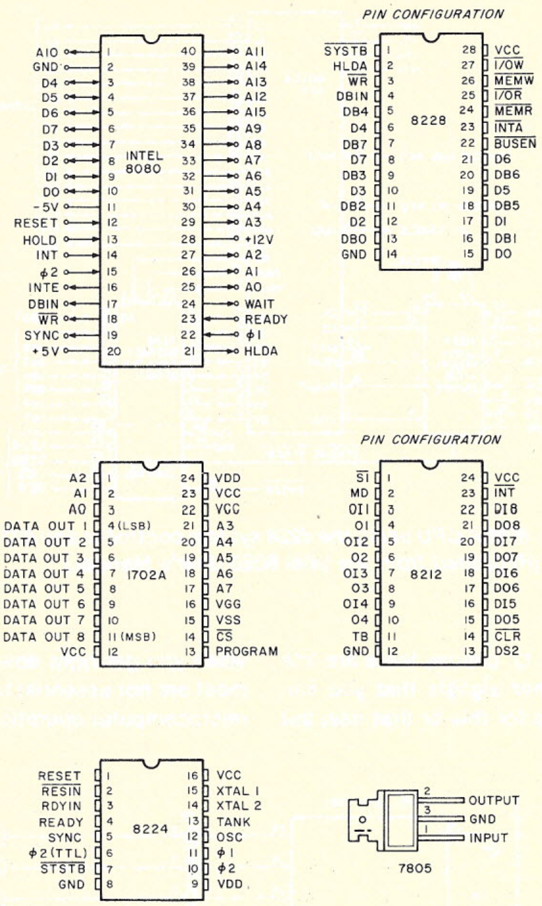



*Fig. 3b. Pin-outs for ICs used in the Simple Computer.*


*A home-constructed computer need not be ugly. This one uses a single enclosure for keyboard, power supply, TV/cassette board, CPU and 24K of static memory plus a 1K monitor memory board. It also has three additional slots.*

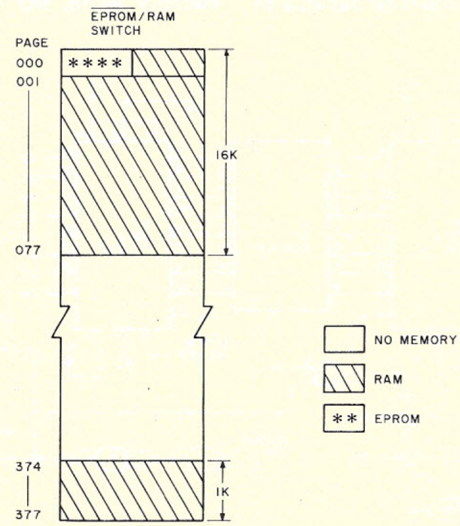



*Fig. 4. A memory map of the system. The 1K is set off so the monitor will be resident regardless of the program in the lower 16K.*

To implement this simplified CPU idea, Intel devised the 8228 system controller. Fig. 2 is a schematic diagram of an 8080A system using both the 8228 controller and 8224 clock chips. This chip set makes available the signals mentioned above plus a few more for special applications. The only deviation from the original description is that the control signals are inverted. This is done to provide simpler interfacing with Intel memory and I/O chips.

The two-way, or bidirectional, data bus may be a little strange to those familiar with the Altair system, in which the data bus is split into a separate input bus (DI0 to DI7) and output bus (DO0 to DO7). The 8080A uses only one or the other at a time, never both. Clearly, then, it should be possible to use only one data bus, as long as input and output conditions are properly controlled. The 8228 accomplishes this task, thus creating a single bidirectional data bus.

A possible limitation of the 8228-based system is that it does not have the data-bus drive capability of other designs. To avoid loading problems you should use only fully buffered peripheral boards. This is not a severe limitation since most hobby-grade boards are buffered to present only one low-power TTL load to the bus. My system has had more

**IN BIT #**

| | |
|---|---|
| 0 | Cassette read (software serial conversion). |
| 1 | Data ready flip-flop (Keyboard): 0 active. |
| 2 | |
| . | Not used |
| . | |
| . | |
| 7 | |

**OUT BIT #**

| | |
|---|---|
| 0 | Cassette write (software serial conversion). |
| 1 | Used to set data ready flip-flop after data input complete. |
| 2 | |
| . | Not used |
| . | |
| . | |
| 7 | TTY Output |

*Fig. 5a. Specifications of control channel.*

than 25 low-power TTL loads on the data bus with no ill effects. Scoping the data lines under this loading revealed no signal degradation.

Fig. 3a shows the full schematic of my CPU board. Disregarding the EPROM and I/O circuits for a moment, note that the design is essentially the same as that in Fig. 2. I have added bus drivers to the address and control lines. The latter could be eliminated in a small system where loading was limited to ten or fewer low-power TTL loads. Here are a few additional comments on the design.

1. Where applicable, bus connections conform to the Altair standard. This does not mean that the bus is directly Altair compatible, though. I have used several Altair-type boards on the bus, but it was sometimes necessary to modify them slightly or bring out additional signals. Note in particular that since my CPU uses a bidirectional data bus, it was necessary to parallel DI0-DI7 and DO0-DO7.

2. A single-level interrupt can be used by connecting pin 14 of the 8080A to the appropriate circuitry.

3. HOLD (pin 13) is grounded and not used, though it could be if desired.

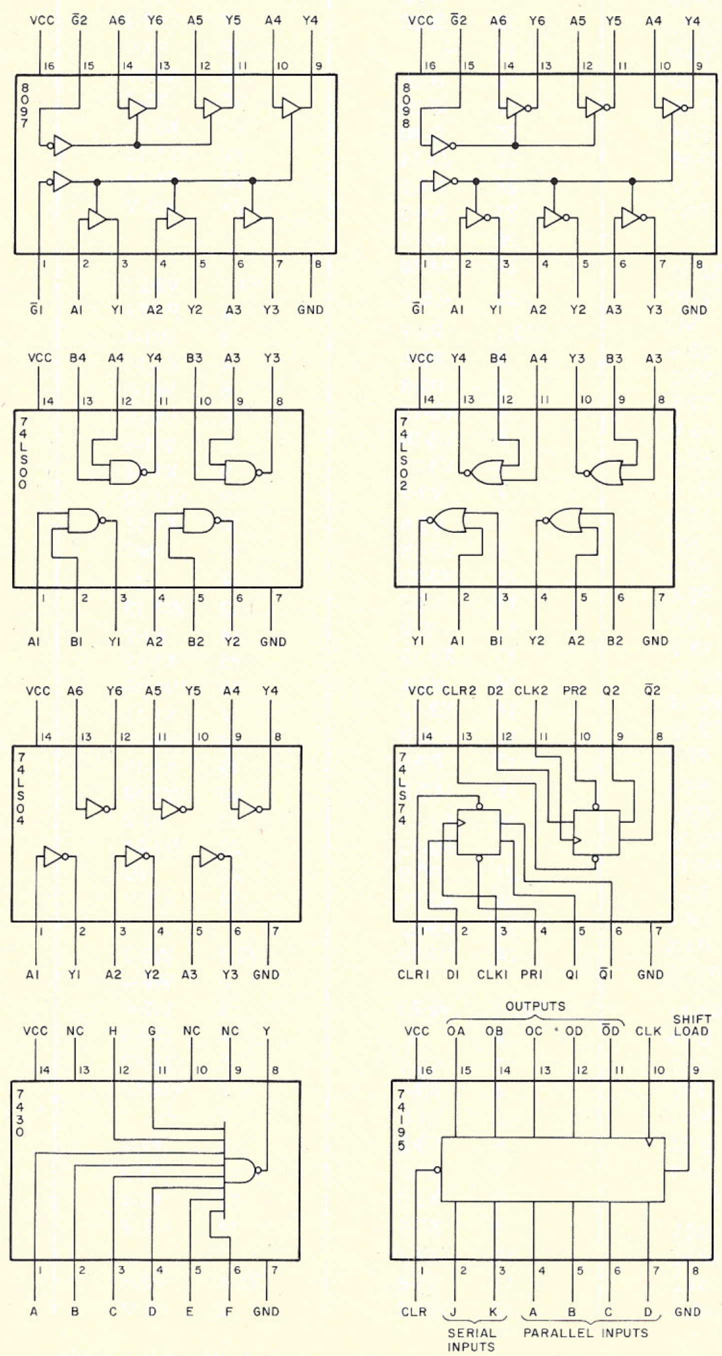
4. The interrupt-enable and processor-wait outputs (pins 16 and 24, respectively) are not used, but with proper buffering they could drive LEDs.

5. The 7404 (ICM4) is used to drive four LEDs to indicate the status of control signals MR, MW, I/OR and I/OW. This additional circuitry could be eliminated, but whoever heard of a computer without blinking lights?!

Another decision I made during the design phase was that there would be no front panel. This meant system start-up would have to be done with an EPROM stored program. Rather than include full monitor software, I elected to use only a cassette loader on EPROM, which provided the opportunity to select and use different monitors without having to reprogram the entire EPROM. Fig. 4 shows a memory map of my system. Note that the EPROM resides at the bottom of memory only during start-up; otherwise, RAM occupies the lowest page. This is important since many available software packages require RAM throughout the lower 16K of memory. A toggle switch is used to select either the EPROM or RAM at page zero. The switch actually controls a logic signal on the bus that I

| LABEL | CODE | COMMENT |
|---|---|---|
| INPUT | IN 002 | GET READY STATUS FROM CONTROL CHANNEL |
| | ANI 002 | LOOK FOR ZERO ON BIT |
| | JNZ INPUT | 2, OTHERWISE DO AGAIN |
| | IN 004 | KEYBOARD DATA INTO A |
| | PUSH SW | SAVE A |
| | MVI A, 3 | SET READY FLIP-FLOP |
| | OUT 002 | AND SET CASSETTE TO MARK TONE |
| | POP SW | RESTORE KEYBOARD DATA |
| | RET | RETURN TO CALLING PROGRAM |

*Fig. 5b. Typical keyboard entry routine.*




*Diagrams of various IC circuits.*

call $\overline{\text{EPROM}}$/RAM. A logic zero on this line selects the EPROM, while a logic one selects RAM.

Bringing up the system from a cold start involves the following steps:

1. Set $\overline{\text{EPROM}}$/RAM switch to $\overline{\text{EPROM}}$.

2. Start cassette containing monitor program data.

3. Hit system reset to start cassette loader program in the EPROM.

4. The monitor is read from the cassette into the top 1K of RAM (pages 374 to 377). At the

| | | | |
|---|---|---|---|
| NC | = not connected. | ⏚* | = use #22 wire directly to ground B-50 or B-100. |
| B | = bus. | R/C | = resistor and/or capacitor (see Fig. 3). |
| V | = voltage source. | XTAX | = 18 MHz Crystal (see Fig. 3). |
| +5 V | = component side lands. | LED | = Light Emitting Diode-cathode. |
| ⏚ | = ground lands on reverse side. | | |

| | |
|---|---|
| S2-1 | P3-14 |
| 2 | ⏚ |
| 3 | V3-6 |
| 4 | V3-19 |
| 5 | V3-21 |
| 6 | V3-8 |
| 7 | V3-10 |
| 8 | V3-12 |
| 9 | V3-17 |
| 10 | V3-15 |
| 11 | −5 V |
| 12 | T2-1 |
| 13 | ⏚ |
| 14 | S2-13 |
| 15 | T2-10 |
| 16 | NC |
| 17 | V3-4 |
| 18 | V3-3 |
| 19 | T2-5 |
| 20 | +5 V |
| 21 | V3-2 |
| 22 | T2-11 |
| 23 | T2-4 |
| 24 | NC |
| 25 | P2-2 |
| 26 | P2-4 |
| 27 | P2-6 |
| 28 | +12 V |
| 29 | P2-10 |
| 30 | P2-14 |
| 31 | P2-12 |
| 32 | P3-2 |
| 33 | P3-4 |
| 34 | P3-6 |
| 35 | P3-10 |
| 36 | P4-10 |
| 37 | P4-2 |
| 38 | P4-4 |
| 39 | P4-6 |
| 40 | P3-12 |
| T2-1 | N4-12 |
| 2 | B-75 |
| 2 | S1-1 |
| 2 | R/C |
| 3 | Y4-1 |
| 6 | B-24 |
| 7 | V3-1 |
| 8 | ⏚ |
| 9 | +12V |
| 14 | XTAL |
| 15 | R/C |
| 16 | +5 V |

| | |
|---|---|
| V3-5 | B-38 |
| 7 | B-90 |
| 9 | B-89 |
| 11 | B-88 |
| 13 | B-36 |
| 14 | ⏚ |
| 16 | B-35 |
| 18 | B-39 |
| 20 | B-40 |
| 22 | V3-14 |
| 24 | N4-2 |
| 25 | N4-6 |
| 26 | N4-4 |
| 27 | N4-10 |
| 28 | +5 V |
| X3-1 | P2-7 |
| 2 | P2-5 |
| 3 | P2-3 |
| 4 | V3-13 |
| 5 | V3-16 |
| 6 | V3-11 |
| 7 | V3-9 |
| 8 | V3-5 |
| 9 | V3-18 |
| 10 | V3-20 |
| 11 | V3-7 |
| 12 | +5 V |
| 13 | X3-12 |
| 14 | V2-2 |
| 15 | X3-13 |
| 16 | −9 V |
| 17 | P3-5 |
| 18 | P3-3 |
| 19 | P2-11 |
| 20 | P2-13 |
| 21 | P2-9 |
| 22 | X3-15 |
| 23 | X3-22 |
| 24 | X3-16 |
| W1-1 | V3-25 |
| 2 | ⏚ |
| 3 | X1-1 |
| 4 | X3-4 |
| 5 | X1-2 |
| 6 | X3-5 |
| 7 | X1-3 |
| 8 | X3-6 |
| 9 | X1-4 |
| 10 | X3-7 |
| 11 | X1-8 |
| 12 | W1-2 |

| | |
|---|---|
| 13 | X3-1 |
| 14 | Y1-6 |
| 15 | X3-8 |
| 16 | X1-5 |
| 17 | X3-9 |
| 18 | X1-6 |
| 19 | X3-10 |
| 20 | X1-7 |
| 21 | X3-11 |
| 22 | W1-12 |
| 23 | P4-14 |
| 24 | +5 V |
| P1-1 | V3-27 |
| 2 | P1-24 |
| 3 | W1-4 |
| 4 | X1-16 |
| 5 | W1-6 |
| 6 | X1-15 |
| 7 | W1-8 |
| 8 | X1-14 |
| 9 | W1-10 |
| 10 | X1-13 |
| 11 | NC |
| 12 | ⏚ |
| 13 | W1-13 |
| 14 | P1-2 |
| 15 | X1-12 |
| 16 | W1-15 |
| 17 | X1-11 |
| 18 | W1-17 |
| 19 | X1-10 |
| 20 | W1-19 |
| 21 | X1-9 |
| 22 | W1-21 |
| 23 | NC |
| 24 | +5 V |
| Y3-1 | W2-2 |
| 2 | W2-4 |
| 3 | W2-6 |
| 4 | W2-8 |
| 5 | W2-10 |
| 6 | W2-12 |
| 7 | ⏚ |
| 8 | Y4-9 |
| 11 | Y2-2 |
| 12 | Y2-4 |
| 14 | +5 V |
| X2-1 | V2-8 |
| 2 | +5 V |

| | |
|---|---|
| 3 | X2-2 |
| 4 | X2-3 |
| 5 | X2-4 |
| 6 | X2-5 |
| 7 | X2-6 |
| 8 | ⏚ |
| 9 | X2-7 |
| 10 | V2-4 |
| 14 | Y4-6 |
| 16 | +5 V |
| Y4-2 | Y4-4 |
| 3 | Y2-8 |
| 5 | X3-14 |
| 7 | ⏚ |
| 8 | B-67 |
| 10 | V2-5 |
| 11 | V2-6 |
| 12 | V3-24 |
| 13 | V2-1 |
| 14 | +5 V |
| Y1-1 | N4-7 |
| 2 | X3-2 |
| 3 | P4-15 |
| 4 | X3-5 |
| 5 | Y1-11 |
| 7 | ⏚ |
| 8 | Y1-12 |
| 9 | N4-9 |
| 10 | Y1-2 |
| 11 | S1-3 |
| 12 | Y1-13 |
| 14 | +5 V |
| S1-1 | S1-4 |
| 2 | P1-3 |
| 3 | S1-11 |
| 4 | S1-10 |
| 5 | B-60 |
| 7 | ⏚ |
| 9 | B-61 |
| 10 | S1-13 |
| 12 | P1-22 |
| 14 | +5 V |
| V2-3 | T2-6 |
| 7 | ⏚ |
| 9 | T2-5 |
| 14 | +5 V |
| W2-1 | P4-9 |
| 3 | P4-7 |
| 5 | P4-5 |
| 7 | ⏚ |

| | |
|---|---|
| 9 | P4-3 |
| 11 | P3-11 |
| 13 | P3-13 |
| 14 | +5 V |
| Y2-1 | P3-9 |
| 3 | P3-7 |
| 7 | ⏚ |
| 9 | B-72 |
| 9 | R/C |
| 14 | +5 V |
| P2-1 | ⏚ |
| P2-3 | B-79 |
| 5 | B-80 |
| 7 | B-81 |
| 8 | ⏚* |
| 9 | B-31 |
| 11 | B-29 |
| 13 | B-30 |
| 15 | P2-1 |
| 16 | +5 V |
| P3-1 | ⏚ |
| 3 | B-82 |
| 5 | B-83 |
| 7 | B-84 |
| 8 | ⏚* |
| 9 | B-34 |
| 11 | B-87 |
| 13 | B-37 |
| 15 | P3-1 |
| P4-1 | ⏚ |
| 3 | B-33 |
| 5 | B-85 |
| 7 | B-86 |
| 8 | ⏚* |
| 9 | B-32 |
| 11 | V3-13 |
| P4-12 | B-59 |
| P4-13 | V3-16 |
| P4-16 | +5 V |
| N4-1 | ⏚ |
| 3 | B-47 |
| 5 | B-68 |
| 7 | B-46 |
| 8 | ⏚* |
| 9 | B-45 |
| 11 | B-99 |
| 15 | N4-1 |
| 16 | +5 V |
| M4-1 | N4-3 |
| 2 | LED |
| 3 | N4-5 |
| 4 | LED |
| 5 | N4-7 |
| 6 | LED |
| 7 | ⏚ |
| 8 | LED |
| 9 | N4-9 |

*Table 1. Wire-wrap guide.*

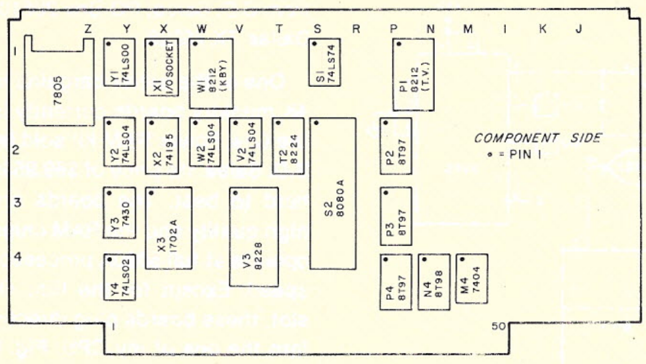



*Fig. 6. A layout diagram for the Vector 8800V prototyping board.*

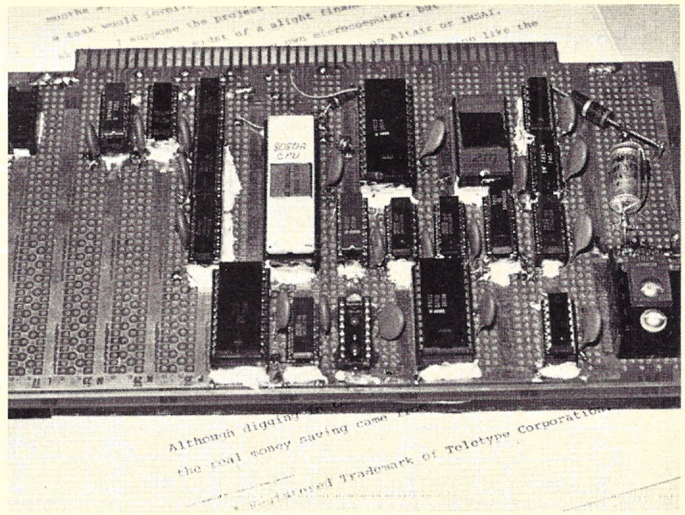
end of data on the cassette, the EPROM program branches to the start of the monitor (374/000).

5. Set $\overline{\text{EPROM}}$/RAM switch to RAM and use the monitor's own cassette routines to load additional software into the lower 16K of memory.

I am a little hesitant to discuss the I/O interface, which is also located on the CPU board. It is so simple you could easily miss it while examining Fig. 3a. Don't feel bad; you were probably expecting one of those 20 IC, $100+ I/O circuits. Remember, I warned you about making things simple. Let's see how the little miracle was wrought.

First, I needed two 8-bit parallel ports—one for the keyboard and one for the TV interface. Two 8212 I/O port chips would do the job nicely. Next, the channel-addressing problem. Upon execution of an input or output instruction, the 8080A places the channel address (up to eight bits) on both halves of the 16-bit address bus. Conventional designs provide circuits to decode the address, thus establishing up to 256 possible I/O channels. For a small system in which fewer than six or eight I/O channels will ever be needed, it seems unnecessary to apply these decoding techniques. Why not simply use the eight address lines individually to control the I/O interface ports. This is precisely the approach used in my design.

The input port (8212 IC W1) is gated on with $\overline{\text{I/OR}}$ and the single address line A2. This corresponds to a channel address 004 octal. The output port (8212 IC P1) is gated on with $\overline{\text{I/OW}}$ and A2 corresponding again to a channel address of 004 octal. Address line A1 is used to gate the control channel, making its channel address 002 octal. The control channel also is used for the cassette interface and Teletype output. Fig. 5a summarizes the specifications of the control channel. Fig. 5b is an example of the machine code used in a typical keyboard entry routine. That about completes description of the CPU design. Now a few words about construction of the board.



*CPU of Simple Computer using wire-wrap sockets and silicone rubber mounting. Epoxy can by used for less conspicuous mounting.*

## Construction

The CPU board was originally hand-wired and soldered, which, I quickly admit, is not the way to do it. Wire-wrapping is a superior method, as I have discovered lately in building a couple more CPUs. Fig. 6 and Table 1 give IC layout data and pin-to-pin wiring information. Follow these suggestions also:

1. Use the component-side land pattern for the +5 V supply and the bottom land pattern for the ground. Connect despiking capacitors (0.1 uF) from the component to the bottom side at a rate of one capacitor per two TTL chips.

2. A 16-pin DIP socket (X1) and plug is used to carry the keyboard and TV interface connections. The cassette interface connections are made through unused pins on the main bus.

3. Power-supply connections to the 14- and 16-pin ICs are not shown in the schematic. Use the following pin-out data:

| | Vcc | GND |
|---|---|---|
| All 14 pin | 14 | 7 |
| All 16 pin | 16 | 8 |

4. The power-supply ground for the address driver ICs (P2, P3 and P4) should be made with #22 wire directly to pin 50/100 on the bus.

The CPU board is the heart of the system, but other peripheral boards and components are needed. Fig. 7 shows a block diagram of the complete system. The following paragraphs briefly describe the makeup of each block.

*Keyboard.* I have used the KB-6 ASCII keyboards available from Ace Electronics, 5400 Mitcheldale B-8, Houston TX 77092. The KB-6 (positive-strobe version) is priced at $39.95 new and $29.95 used. The keyboard is supplied with

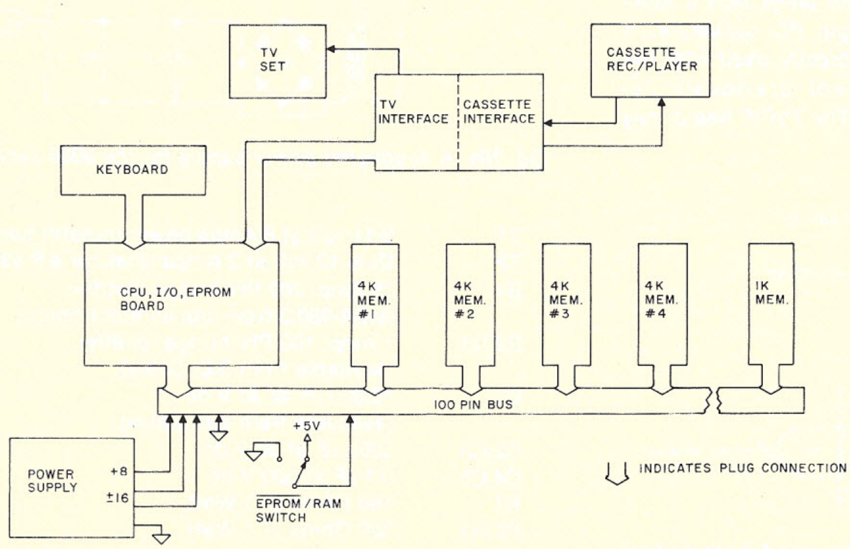



*Fig. 7. A block diagram of the complete system.*

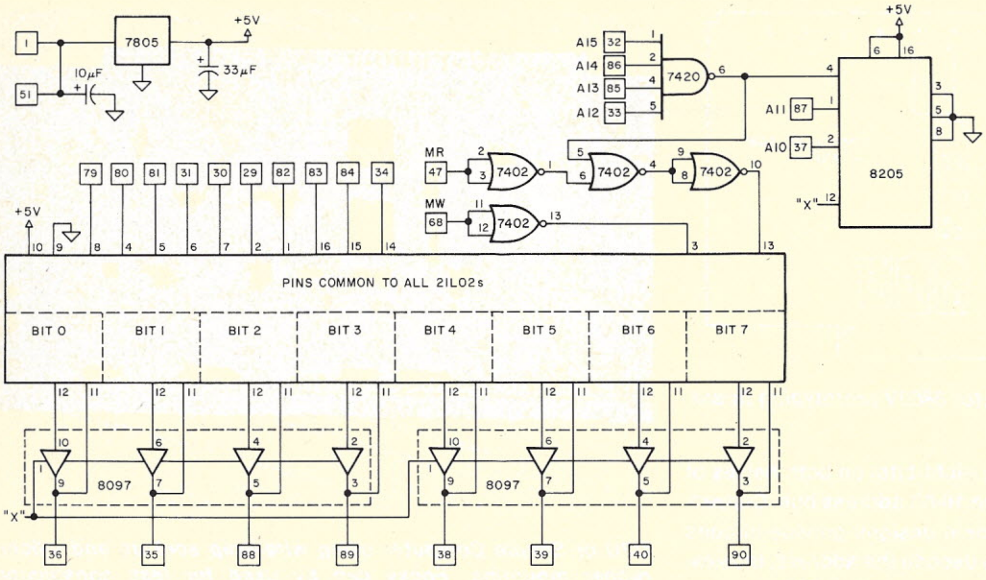



*Fig. 8. The schematic of a memory board to operate at the top 1K position.*

schematic and instructions. It requires an external power supply of +5 and −12 volts. Otherwise, it connects directly to the data and strobe inputs on the CPU board.

*TV/cassette interface.* In my opinion, the best bargain around today is the Digital Group TVC-F board. The $130 kit price includes a versatile 32-character by 16-line TV interface and a 1100-baud cassette interface. The latter uses a Suding audio standard, which I (and a good many others) have found to constitute an effective and reliable data-recording technique. During the past year and a half, I have done extensive software development—ranging from Tiny BASIC Extended to a full disk BASIC—all with the 1100-baud Suding system. I do not have to buy expensive cassette tape; the 3-for-$2 variety is fine.

In addition, exchanging tapes with other Suding users has demonstrated that variations in cassette player quality have little effect on reliability. Although I am aware of the theoretical limitations of the system, it is difficult to argue with success.

The TV-interface portion of the board produces stable high-quality video that includes all printable ASCII characters (lowercase, too!). It has on-board memory that permits it to operate as a parallel 8-bit I/O device. With appropriate software, it can be made to function in a page or scroll mode with or without a cursor.

The TVC-F board requires ±12 volts and +5 volts for power and plugs into a standard 22-pin PC connector. It can be directly used with the CPU board previously described. The TVC-F board may be obtained from The Digital Group, 1031 W. Center Ave., Denver CO 80223.

*Memory boards.* 1K memory: In my system I hand-wired this too. The schematic is shown in Fig. 8. Although I placed it on a separate prototyping board, you might be able to squeeze it onto the CPU board. The eight memory chips can be obtained from S.D. Sales, PO Box 28810, Dallas TX 75228.

One of the better bargains in 4K memory boards currently is the Low Power RAM kit sold by S.D. Sales. Its price of $89.95 is hard to beat. The boards are high quality and the RAM chips operate at full 8080A processor speed. Except for the first 4K slot, these boards plug directly into the bus of my CPU. Fig. 9 shows the modifications necessary to the first 4K board to permit switching it on and off with the EPROM/RAM switch. Inquiries about these boards can be made to S.D. Sales at the above address.

*Motherboard.* There are several alternatives here. I simply bought wire-wrap sockets and connected them by hand. This was a slow but inexpensive way to go (by the way, you would not have to wire all 100 pins—only 40 or so). Vector Graphic sells an 18-slot motherboard for $49 that will do nicely. If you have a friend who also wants a system, why not split the cost, then saw the board in half, making two 9-slot

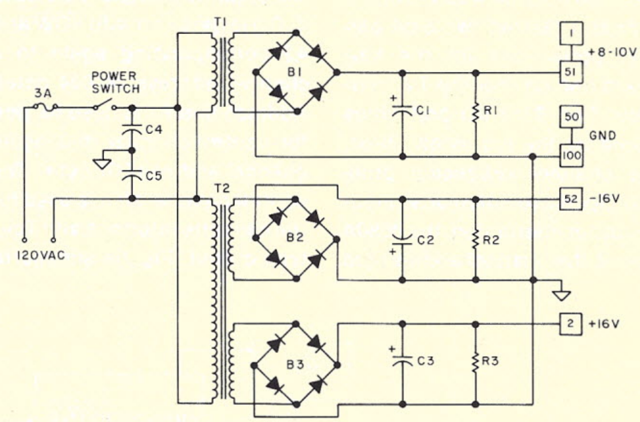



*Fig. 10a. A suggested power supply for the 8080 system.*

| | |
|---|---|
| T1 | 9-11 volt at 8 Amps power transformer. |
| T2 | Dual 12 volt at 2 Amps (Stancor # P-6377). |
| B1 | 25 Amp, 200 PIV bridge rectifier (MDA-980-3 from James Electronics). |
| B2,B3 | 1 Amp, 100 PIV bridge rectifier (available from S.D. Sales). |
| C1 | 40,000 µF at 30 V dc (available from S.D. Sales). |
| C2,C3 | 3000 µF at 30 V dc |
| C4,C5 | 0.1 µF at 600 V dc |
| R1 | 180 Ohms at 1 Watt. |
| R2,R3 | 330 Ohms at 1 Watt. |

*Fig. 10b. Suggested parts list for power supply.*

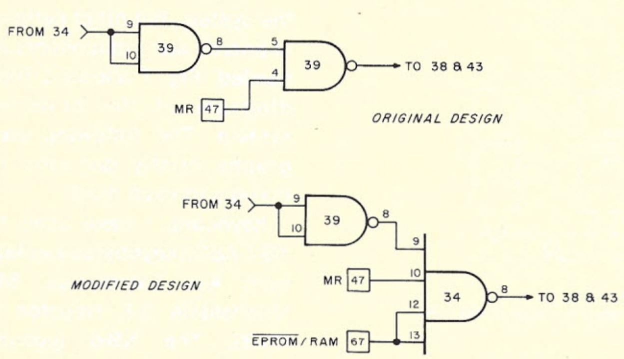



*Fig. 9. Modifications of the S.D. Sales Low Power RAM for the first 4K slot.*

pieces. The 100-pin sockets can be obtained from Godbout Electronics, Box 2355, Oakland Airport CA 94614, at a cost of five for $22.

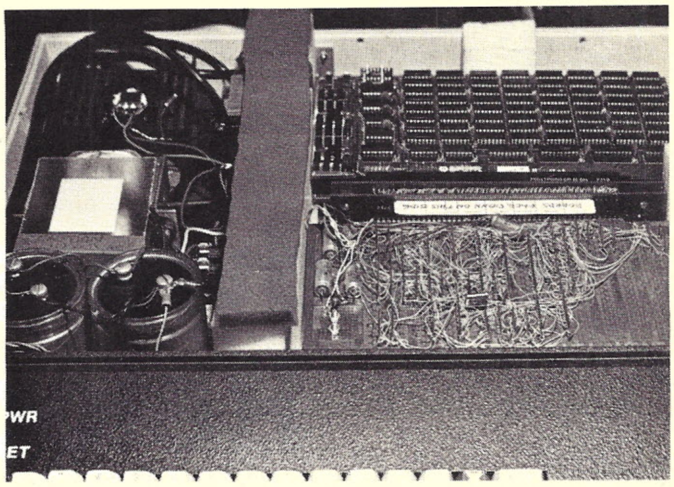
*Chassis and power supply.* The system can be built into an upturned aluminum chassis such as the BUD AC-419 (11 × 17 × 2½ inches). A suitable power supply schematic is shown in Fig. 10. The parts list (Fig. 10b) is only suggested since you may be able to dig these or similar parts out of the junk box and save some money.

*EPROM and software.* I will program a 1702A with the cassette loader for $3 postpaid. A cassette containing the system monitor with appropriate documentation is available for $7.50 postpaid. Other software can be added to the monitor cassette at your request. The following prices are in addition to the base price of $7.50.

1. Monitor with Baudot Printer Driver, add $2.
2. Tiny BASIC Extended, add $7.50.
3. BASIC ETC (8K version), add $20.
4. BASIC ETC (10K version), add $25.

Address correspondence to:

The Simple Computer
Dick Whipple
305 Clemson Dr.
Tyler TX 75703.■




*Power supply and fan are on the left; the vertical motherboard, in two sections, is on the right. Four slots are provided on each side. Wire-wrapped CPU card is on the front. Baffles are used to direct cooling air under fan assistance through the cabinet.*

# Hardware Program Relocation

## Part 2: the software

*Dr. Michael Wingfield*
*93 Pine Hill Road*
*Bedford MA 01730*

In a previous article (*Kilobaud* No. 13, p. 60), I discussed the design of base-register hardware that can be added to a microcomputer to enable relocation of software programs without the need for special assemblers or loaders. However, before the new hardware architecture can be used most effectively, an understanding of associated software issues is necessary. This article will describe a set of system subroutines that enable programs with one base-register value to communicate with programs or data utilizing another.

### Background

The base-register (BR) hardware intercepts the eight higher-order address-bus lines coming from the microprocessor and produces an eight-bit output that is the sum of the BR contents and the microprocessor unit (MPU) address bus. The lower eight bits of the address bus are not modified. The memory and peripheral chips see only the modified or effective address as shown in Fig. 1. Thus, an MPU reference to 100 hex with a BR value of 12 hex produces the value 1300 hex on the address bus.

The microprocessor thinks it is referencing location 100 hex, and instead retrieves/stores location 1300 hex. With this hardware, a program originated at location zero can be relocated anywhere in memory, as long as it begins on a 256-byte page boundary, and can execute correctly by properly setting the BR.

### Software Issues

As long as a program references its own data and does not transfer control outside of itself, no special system software is required. However, there are operations a program may need to do that require special subroutines. For example, a relocatable software module cannot change its own base register. An attempt to do so will cause the effective program counter to point to the wrong place in memory for the next instruction fetch.

There are operations a relocatable program may wish to do: (1) load/store a value in the data space of another relocatable program; (2) jump to a place in a relocated program; (3) call a relocated program. Knowledge of the destination entry point is

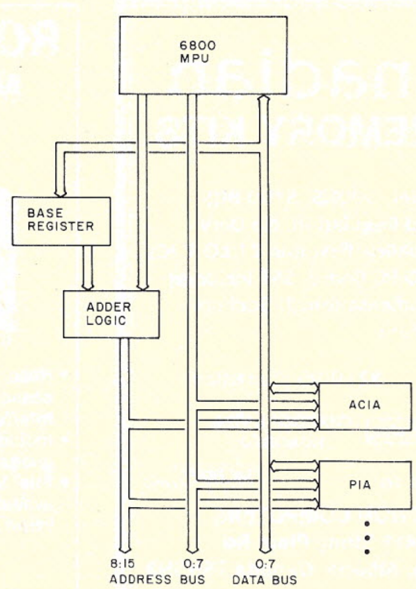



*Fig. 1. System architecture.*

assumed to be known only in terms of the base-register value of the destination program and the entry offset address.

Also at issue is how to handle interrupts and interrupt returns when the interrupt handler is a relocated program having a different BR value than the program interrupted. Furthermore, system resets must be properly handled. The remainder of this article describes a set of subroutines, called the kernel, that handles each of the operations just described.

### Memory Layout

Fig. 2. shows the layout of the upper 4K of memory dedicated to peripheral addresses and kernel software in my system. Recall from the first article that the base-register hardware was designed to permit address transparency when the address bus is pointing to the upper 4K of memory. In other words, when the MPU issues an address in the range F000 hex to FFFF hex, the address is not modified. This was specifically included to permit software executing in that region to change the base register. In this way, a sudden change in BR value has no impact on sequential instruction execution.

A system stack 1K bytes long, located in F000 to F3FF hex, serves all developed programs, which use it freely for variable storage, subroutine returns and interrupt stack. The

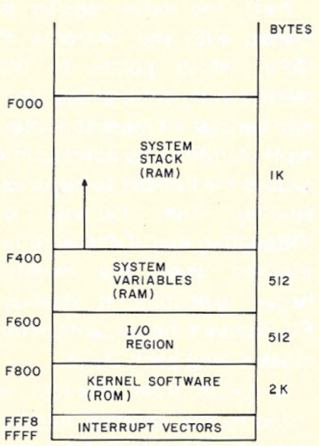



*Fig. 2. System address space allocation.*

location of the stack in the upper 4K region insures that the stack pointer address will not be modified by the BR. Another area of 512 bytes of RAM is set aside for system variables and a common region for programs, and occupies location F400 to F5FF hex. The peripheral-device region is located between F600 and F7FF hex and is where the I/O devices such as asynchronous chips, PIA chips, BR and other devices are found. Locations F800 to FFF7 hex are reserved for ROMs and hold the privileged kernel routines, which I will discuss in the next sections. Finally, the interrupt vectors are located in the last eight locations in memory.

### System Reset

The first operation of a microcomputer, after power is applied, is to enter the system reset code. In the 6800, a reset causes a trap to location FFFE hex, where a sixteen-bit address of the system reset code is found. First, the base register must be initialized to the starting page of the monitor software.

Referring to the kernel listing, in the software starting at SysReset, the base page is read from a location StartPage. In my system, this is an eight-bit hardware switch block made from dual in-line switches and additional logic that interfaces it to the system bus. This permits the starting page to be externally modified without the ROM which contains the SysReset code having to be changed.

This instruction could be changed to an immediate mode if the relocatable monitor code is always in a fixed location. Fig. 3 shows the hardware that can be used to interface a set of switches to the 6800 microcomputer bus. When the MPU loads from location F607 hex, it retrieves the contents of the eight switches.

The system stack is then initialized. In my system, the application programs never initialize the stack pointer and always assume that it is properly set. Finally, the system reset

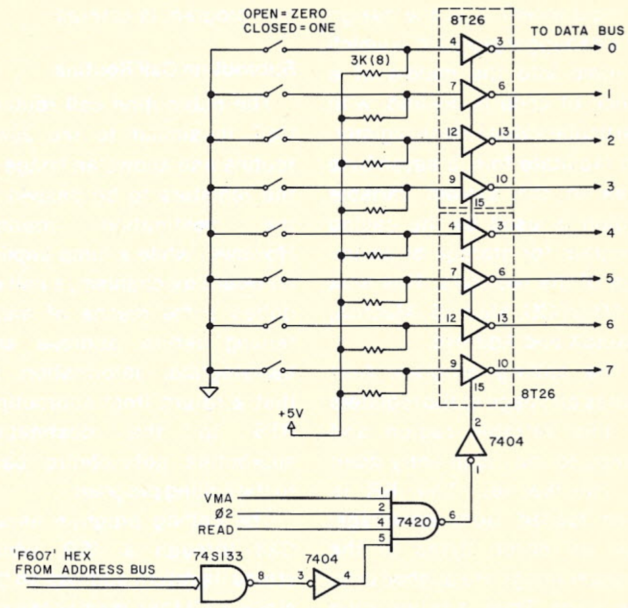



*Fig. 3. Switches interface.*

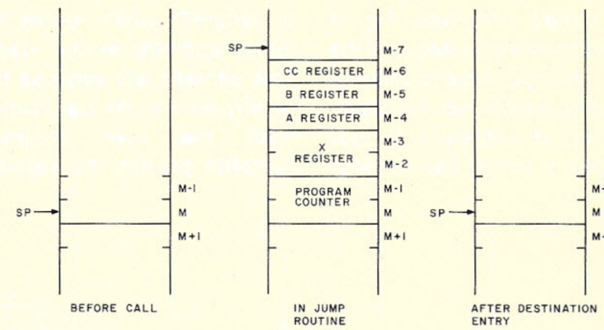



*Fig. 4. Jump stack sequence.*

code does a transfer to location zero, where it is assumed the monitor code begins. If the BR value were 22 hex, the effective starting address would be 2200 hex.

### Store Byte Subroutine

The Store subroutine handles the storing of a byte by a program executing with one BR value into a location defined by a different base register and a relative offset. Remember that normal loads and stores also get their addresses modified by the BR. The executing program puts the destination page in the location BasePage and places the relative offset in the index register X.

A call to the Store subroutine results in the current value of the BR being pushed onto the stack and the destination base being loaded into the BR. The value in register A is then stored, and the original BR is restored before a return is made to the calling program. If the calling program wants to store a byte in an absolute memory location, it must zero BasePage.

### Load Byte Subroutine

The Load subroutine performs nearly identically to the Store subroutine. The base registers are exchanged, and the byte is loaded into register A.

### Jump Routine

To permit a transfer of control from a program using one BR value to another using a different BR value, the Jump routine is entered. This program was designed to permit the passing of parameters to the destination routine via all of the machine registers (i.e., A, B, X and CC). An example of such

a requirement is in the design of a debugger program in which a jump into the middle of a piece of code is desired, with particular values in the register. To facilitate this, a seven-byte area in the system variable region is used by the calling program for storage of an image of the registers. This area is MacroCC, MacroB, MacroA, MacroX and Address.

The calling program first stores an image of the registers in this variable region and jumps to the Jump entry point in the kernel. The BR is then loaded from BasePage, and all seven bytes of the register image are pushed onto the stack. Then, a return from interrupt, RTI, is executed; this simply loads a new machine state from the stack and enters the called program. Fig. 4 shows the various stages of the stack. It is possible to utilize a less elaborate scheme if initialization of registers is not necessary before the destination program is entered.

**Subroutine Call Routine**

The subroutine call routine, Call, is similar to the Jump routine and allows an image of the registers to be passed to the destination routine. However, while a jump implies no return mechanism, a call requires some means of maintaining return address and base-register information so that a return from subroutine, RTS in the destination subroutine, gets control back to the calling program.

The calling program enters Call through a JSR, which stores its return address on the stack. Call then pushes the old BR onto the stack and loads the BR with the destination BR. It then performs a subroutine call to JumpRTI, which stores the return address on the stack. This address will serve as the reentry point to the Call routine from the user program. JumpRTI pushes the register image onto the stack and does an RTI, transferring control to the called program with all of the registers preset. Fig. 5 shows the contents of the stack at various stages of this operation.

The destination subroutine eventually terminates by executing an RTS; this causes a return to CallRet since that address was the last address stored on the stack. The old BR value is pulled from the stack and loaded, and a return is made to the calling program.

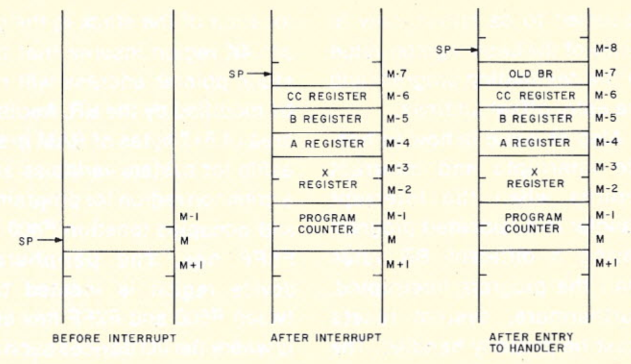



*Fig. 6. Interrupt stack sequence.*

**Interrupt Routines**

Interrupts require special attention because the interrupt handler may also be a relocatable program. In my system, the interrupt routines in the kernel get the handler addresses through RAM locations that are initialized at start-up.

There are three interrupt types on the 6800: the maskable interrupt, which traps to FFF8 hex, the nonmaskable interrupt, which traps to FFFC hex and the software interrupt, which traps to FFFA hex. All three are handled in the same fashion in the kernel software, so I will describe only the maskable-interrupt sequence. An I/O interrupt causes a trap to FFF8 hex and puts the address GoIO, or FFCB hex, in the program counter. GoIO is then entered, which first pushes the old value of the BR on the stack and loads the new value from one previously stored at IOBaseReg.

Next, the index register is loaded with the contents of IOPtr, which points to the relative entry point of the interrupt handler for maskable interrupts. A JMP to the address in X causes the handler to begin executing. The contents of IOBaseReg and IOPtr were initialized when the monitor began executing at start-up. Fig. 6 shows the contents of the stack during each stage.

A return from interrupt is made by the interrupt handler through a jump to CallRet; this pops the old BR off the stack, stores it and executes an RTI,

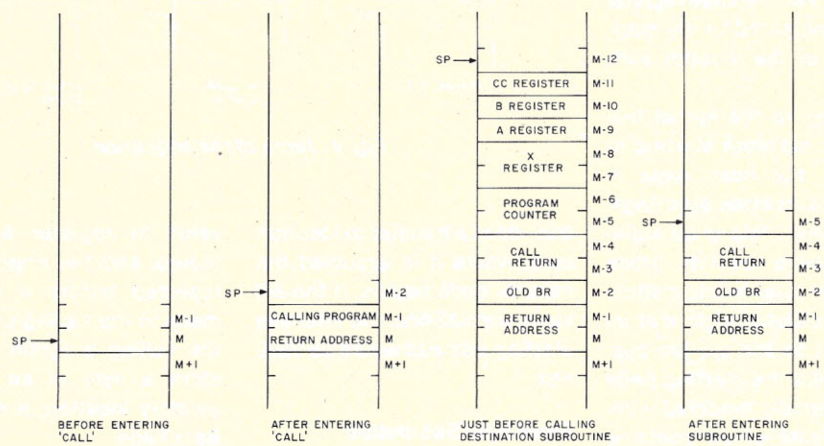



*Fig. 5. Subroutine call stack sequence.*

*Kernel listings.*

```
F3FF   Stack       equ   $f3ff   beginning of stack
F400   BasePage    equ   $f400   new base register value
F401   IOBaseReg   equ   $f401   BR for IO interrupt
F402   IOPtr       equ   $f402   ptr to IO interrupt routin
F404   SwiBR       equ   $f404   BR for swi interrupt
F405   SwiPtr      equ   $f405   ptr to swi interrupt routi
F407   NmiBR       equ   $f407   BR for nmi interrupt
F408   NmiPtr      equ   $f408   ptr to nmi interrupt routi
       *The following are the stack variables for Jump, Call
F410   MacroCC     equ   $f410   place to save CC
F411   MacroB      equ   $f411   place to save reg B
F412   MacroA      equ   $f412   place to save reg A
F413   MacroX      equ   $f413   place to save index reg
F415   Address     equ   $f415   place to save PC
       *end of variables list
```

which restores the MPU to its pre-interrupt state.

**Conclusion**

This concludes a two-part description of the base-register hardware and software I have used with success in my microcomputer. All of the issues considered really deal with an extension of the concept of machine state. The state of a program at any instant can be represented by the contents of the general registers and the value in the program counter. The BR is another state-of-the-system component and must be preserved anytime the other states are preserved, such as during interrupts.

One of the problems in developing software for general distribution to persons having diverse operating systems and hardware configurations—even when all use the same instruction set—is program relocation. It would be nice to buy a ROM containing a desirable program from a computer store and plug it into my system anywhere in memory space and be able to use it. A greatly expanded marketplace would exist if current microcomputers contained program-relocation hardware that, regardless of where the memory is, allowed the owner to plug in ROM chips packaged on a standard board. With ROMs currently able to contain 16K bits, whole applications can be placed in a very small space.

This conjures up an image of little 3 by 5 standard boards, each containing a ROM, in a display rack in a computer store—just like cassettes. They would cost about ten dollars. The user would pick out an interesting one; take it home or to work; plug it in; set up the initialization parameters such as I/O-device routine entry points, RAM area and BR value; and be on the air.

I am currently investigating an architecture that utilizes two base registers—one for program relocation, the other for data relocation. This would permit development of more generalized relocation schemes.■

```
F606   BaseReg          equu   $f606          system base register
F607   StartPage        equ    $f607          switch register

FF74                                org     $ff74
                    *Subroutine for getting a byte from another space
FF74   F6   F606    Load            lda  b  BaseReg     get BR
FF77   36                           psh  a              save current BR
FF78   F6   F400                    lda  b  BasePage    get page number
FF7B   B7   F606                    sta  a  BaseReg     put in BR
FF7E   A6   00                      lda  a  x           read byte
FF80   33                           pul  b              get old BR
FF81   F7   F606                    sta  b  BaseReg     restore calling BR
FF84   39                           rts                 return to calling program
                    *Subroutine for storing a byte in another space
FF85   F6   F606    Store           lda  b  BaseReg     get BR
FF88   37                           psh  b              save current BR
FF89   F6   F400                    lda  b  BasePage    get new page
FF8C   F7   F606                    sta  b  BaseReg     put in BR
FF8F   A7   00                      sta  a  x           store the byte
FF91   33                           pul  b              get old BR
FF92   F7   F606                    sta  b  BaseReg     restore calling BR
FF95   39                           rts                 return to calling program
                    *Routine for jumping to program in another space
FF96   B6   F400    Jump            lda  a  BasePage    get page #
FF99   B7   F606                    sta  a  BaseReg     put it in BR
FF9C   C6   07      JumpRTI         lda  b  #7          index value
FF9E   CE   F416                    ldx     #Address+1
FFA1   A6   00      PushLoop        lda  a  x           put values on stack
FFA3   36                           psh  a
FFA4   09                           dex
FFA5   5A                           dec  b
FFA6   26   F9                      bne     PushLoop
FFA8   3B                           rti                 load new machine state
                    *Subroutine for calling a subroutine in another space
FFA9   B6   F606    Call            lda  a  BaseReg     get old base reg
FFAC   36                           psh  a              save on stack
FFAD   B6   F400                    lda  a  BasePage    get page #
FFB0   B7   F606                    sta  a  BaseReg     put it in BR
FFB3   8D   E7                      bsr     JumpRTI     save return addr
                    *Return is made here
FFB5   32           CallRet         pul  a              get Base Reg
FFB6   B7   F606                    sta  a  BaseReg
FFB9   39                           rts                 return to calling prog
                    *Routine for continuing a program
FFBA   32           IntReturn       pul  a              get BR off stack
FFBB   B7   F606                    sta  a  BaseReg
                    *SP assumed to point to interrupt frame
FFBE   3B                           rti                 return to interrupted prog
                    *System reset code
FFBF   B6   F607    SysReset        lda  a  StartPage   Hardware byte
FFC2   B7   F606                    sta  a  BaseReg     inz base register
FFC5   8E   F3FF                    lds     #Stack      inz stack pointer
FFC8   7E   0000                    jmp     0           start up OS
                    *Interrupt routines
FFCB   B6   F606    GoIO            lda  a  BaseReg
FFCE   36                           psh  a              save old BR
FFCF   B6   F401                    lda  a  IOBaseReg
FFD2   B7   F606                    sta  a  BaseReg     get IO BR
FFD5   FE   F402                    ldx     IOPtr
FFD8   6E   00                      jmp     x           go to IO interrupt
                    *Software interrupt routine
FFDA   B6   F606    GoSwi           lda  a  BaseReg
FFDD   36                           psh  a              save old BR
FFDE   B6   F404                    lda  a  SwiBR
FFE1   B7   F606                    sta  a  BaseReg
FFE4   FE   F405                    ldx     SwiPtr
FFE7   6E   00                      jmp     x           go to swi interrupt
                    *Nonmaskable interrupt routine
FFE9   B6   F606    GoNmi           lda  a  BaseReg     get current BR
FFEC   36                           psh  a              save it
FFED   B6   F407                    lda  a  NmiBR       get new BR
FFF0   B7   F606                    sta  a  BaseReg
FFF3   FE   F408                    ldx     NmiPtr
FFF6   6E   00                      jmp     x           go to nmi interrupt
                    *Interrupt vectors
FFF8        FFCB    IOInt           fdb     GoIO
FFFA        FFDA    SwiInt          fdb     GoSwi
FFFC        FFE9    NmiInt          fdb     GoNmi
FFFE        FFBF    Reset           fdb     SysReset
                                    end
```

*Dave Alverson*
*8717 Empire CT.*
*Cincinnati OH 45231*

# State Capitals

## *a new education program for the kids*

This educational program will show off to your friends the practical applications of your microcomputer. The user must match a state with its capital by filling in the answer or picking one of four choices. Running the program is simple, as seen by the sample run; here's how it works.

First, the program allocates memory space in line 120 for four arrays. The numeric A array contains information for each state, and a record of whether a given state and its capital have been correctly matched. Since Alabama is first alphabetically, A(1) contains its data. When A(1) is zero, the state and its capital (or vice versa) have not been matched yet. When the value is one it was matched incorrectly, and when it is two the correct answer has been given. The numeric B array is used only when multiple-choice questions are asked. B(1) is the state number (1 to 50) of the first choice of the four. The C$ and S$ arrays contain the state and capital character strings. C$(2) is the capital (Juneau) of the second state, S$(2), Alaska.

In lines 140 and 150, the C$ and S$ arrays are read in from the data statements at the end of the program. Line 160 sets the 50 elements of the A array equal to zero. The total correct, N, and the number of guesses, G, are also set to zero. This is necessary when you want to go through the questions a second time.

In lines 170 through 290, two variables are set which determine how the questions are to be asked. When X equals one, the user must answer from four choices. When X equals two, the user must type in the answer with correct spelling. The state is given and the capital is asked for when Y equals one. If Y is two, the state is requested and the capital is given.

The loop starting at line 320 is used to pick a number from 1 to 50 representing a state that has not been successfully matched. This number is assigned to the variable R. The loop will execute up to ten times, each time picking a random number and checking if that state has been matched correctly. If not, the program will exit the loop and get ready to ask the question. If it has been answered, the program will go back to pick another state.

A second loop, starting at line 370, is used if an unmatched state was not picked in the ten tries of the first loop. This second loop takes the first state, going from 1 to 50, that has not been matched.

Line 400 will print that statement if the question was answered incorrectly the last time it was asked. Execution of the program goes to the multiple-choice section if X equals one in line 410. The

*Program listing.*

```
100 REM STATES AND CAPITALS QUIZ PROGRAM
110 REM BY DAVE ALVERSON JULY, 1977
120 DIM A(50),B(4),C$(50),S$(50)
130 REM READ IN STATE AND CAPITAL ARRAYS
140 FOR I=1 TO 50
150 READ S$(I),C$(I) : NEXT I
160 FOR I=1 TO 50 : A(I)=0 : NEXT I : G=0 : N=0
170 PRINT "YOU HAVE YOUR CHOICE OF FILL-IN OR MULTIPLE CHOICE"
180 INPUT "WOULD YOU LIKE TO FILL-IN THE ANSWERS";Z$
190 IF Z$="Y" OR Z$="YES" THEN 230
200 PRINT "MULTIPLE CHOICE — ANSWER EACH QUESTION WITH 1,2,3 OR 4"
210 X=1 : PRINT "TO STOP TYPE 0 (ZERO) FOR YOUR ANSWER"
```

next few lines print the fill-in question and set A$ equal to the correct answer.

Line 500 says if the length of the input is only one character then check if you want to stop. If the answer is correct, the program goes to line 770; if not, it goes to line 760.

The multiple-choice section starts with line 540. This line picks a random integer from one to four. The variable C represents the number of the correct choice. The loop at line 560 picks four state numbers and puts them in the B array. Line 590 sets B with a subscript of C, B(C), equal to the correct state number, R. If any of the four choices is the same, line 610 or 620 will send all four back and pick four choices again. The next ten lines print out the choices, either states or capitals, and ask the question. If the input is out of the one-to-four range, then it checks if you want to stop.

If you are wrong, line 760 sets A(R) equal to one, which means you answered incorrectly. When you answer correctly the number correct, N, is incremented, A(R) is set to two (meaning you gave the right answer), and it prints out your total correct. The variable G, which is the total number of guesses, is incremented. If the number of states correctly matched is less than 50, it goes back to ask another question. If you have matched all 50, then it goes down to print your totals and check if you want to try again.

Lines 810 to 840 check to see if you want to stop. Program execution comes to these lines when the fill-in input is one character long or the multiple-choice input is not in the one-to-four range. If you do not want to stop, the program asks for your answer and goes to the input line—730 for multiple choice, 490 for fill-in. The four lines starting at 850 are used when you want to stop or after you have answered all

```
220 GOTO 250
230 PRINT "FILL-IN — YOU MUST SPELL EXACTLY! (SAINT IS ABBREVIATED ST.)"
240 X=2 : PRINT "TO STOP TYPE S FOR YOUR ANSWER"
250 PRINT
260 PRINT "YOU HAVE YOUR CHOICE OF WHETHER THE STATE OR CAPITAL IS ASKED"
270 INPUT "WOULD YOU LIKE TO ANSWER WITH THE CAPITAL";Z$
280 Y=2
290 IF Z$="Y" OR Z$="YES" THEN Y=1
300 RANDOMIZE : PRINT : PRINT
310 REM PICK A STATE
320 FOR I=1 TO 10
330 R=INT(RND*50)+1
340 IF A(R) <> 2 THEN 400
350 NEXT I
360 REM DON'T WASTE TIME PICKING ONE
370 FOR R=1 TO 50
380 IF A(R) <> 2 THEN 400
390 NEXT R : GOTO 850
400 IF A(R)=1 THEN PRINT "TRY THIS ONE AGAIN"
410 IF X=1 THEN 520
420 REM THIS SECTION ASKS FOR FILL-IN ANSWERS
430 IF Y=2 THEN 470
440 A$=C$(R)
450 PRINT "WHAT IS THE CAPITAL OF";S$(R);
460 GOTO 490
470 A$=S$(R)
480 PRINT C$(R);" IS THE CAPITAL OF";
490 INPUT Z$
500 IF LEN(Z$)=1 THEN 810
510 IF Z$=A$ THEN GOTO 770 ELSE GOTO 760
520 REM THIS SECTION ASKS MULTIPLE CHOICE QUESTIONS
530 REM THE VALUE OF C IS THE CORRECT ANSWER
540 C=INT(RND*4)+1
550 REM PICK FOUR STATES FOR THE CHOICES
560 FOR I=1 TO 4
570 B(I)=INT(RND*50)+1
580 NEXT I
590 B(C)=R
600 REM MAKE SURE NONE ARE THE SAME
610 IF B(1)=B(2) OR B(1)=B(3) OR B(1)=B(4) THEN 560
620 IF B(2)=B(3) OR B(2)=B(4) OR B(3)=B(4) THEN 560
630 IF Y=2 THEN 690
640 REM PRINT CAPITAL CHOICES
650 PRINT "1. ";C$(B(1));TAB(20);"3. ";C$(B(3))
660 PRINT "2. ";C$(B(2));TAB(20);"4. ";C$(B(4))
670 PRINT "THE CAPITAL OF ";S$(R);" IS";
680 GOTO 730
690 REM PRINT STATE CHOICES
700 PRINT "1. ";S$(B(1));TAB(20);"3. ";S$(B(3))
710 PRINT "2. ";S$(B(2));TAB(20);"4. ";S$(B(4))
720 PRINT C$(R);" IS THE CAPITAL OF";
730 INPUT Z : Z=INT(ABS(Z))
740 IF Z <1 OR Z >4 THEN 810
750 IF Z=C THEN 770
760 A(R)=1 : PRINT "WRONG" : GOTO 790
770 A(R)=2 : N=N+1
780 PRINT "RIGHT! YOU HAVE ";N;"CORRECT"
790 PRINT : G=G+1
800 IF N <50 THEN GOTO 310 ELSE GOTO 850
810 INPUT "DO YOU WANT TO STOP";Z$
820 IF Z$="Y' OR Z$="YES" THEN 850
830 PRINT "YOUR ANSWER FOR LAST QUESTION";
840 ON X GOTO 730, 490
850 PRINT
860 PRINT "YOU GOT ";N;"RIGHT IN ";G;"GUESSES"
870 INPUT "WOULD YOU LIKE TO TRY AGAIN";Z$
880 IF Z$="Y" OR Z$="YES" THEN PRINT : GOTO 160
890 DATA "ALABAMA","MONTGOMERY","ALASKA","JUNEAU","ARIZONA"
892 DATA "PHOENIX","ARKANSAS","LITTLE ROCK","CALIFORNIA"
894 DATA "SACRAMENTO","COLORADO","DENVER","CONNECTICUT","HARTFORD"
896 DATA "DELAWARE","DOVER","FLORIDA","TALLAHASSEE","GEORGIA"
898 DATA "ATLANTA","HAWAII","HONOLULU","IDAHO","BOISE","ILLINOIS"
900 DATA "SPRINGFIELD","INDIANA","INDIANAPOLIS","IOWA"
902 DATA "DES MOINES","KANSAS","TOPEKA","KENTUCKY","FRANKFORT"
904 DATA "LOUISIANA","BATON ROUGE","MAINE","AUGUSTA","MARYLAND"
906 DATA "ANNAPOLIS","MASSACHUSETTS","BOSTON","MICHIGAN"
908 DATA "LANSING","MINNESOTA","ST. PAUL","MISSISSIPPI","JACKSON"
910 DATA "MISSOURI","JEFFERSON CITY","MONTANA","HELENA","NEBRASKA"
912 DATA "LINCOLN","NEVADA","CARSON CITY","NEW HAMPSHIRE"
914 DATA "CONCORD","NEW JERSEY","TRENTON","NEW MEXICO","SANTA FE"
916 DATA "NEW YORK","ALBANY","NORTH CAROLINA","RALEIGH"
918 DATA "NORTH DAKOTA","BISMARCK","OHIO","COLUMBUS","OKLAHOMA"
920 DATA "OKLAHOMA CITY","OREGON","SALEM","PENNSYLVANIA"
922 DATA "HARRISBURG","RHODE ISLAND","PROVIDENCE","SOUTH CAROLINA"
924 DATA "COLUMBIA","SOUTH DAKOTA","PIERRE","TENNESSEE"
926 DATA "NASHVILLE","TEXAS","AUSTIN","UTAH","SALT LAKE CITY"
928 DATA "VERMONT","MONTPELIER","VIRGINIA","RICHMOND","WASHINGTON"
930 DATA "OLYMPIA","WEST VIRGINIA","CHARLESTON","WISCONSIN"
932 DATA "MADISON","WYOMING","CHEYENNE"
950 END
```

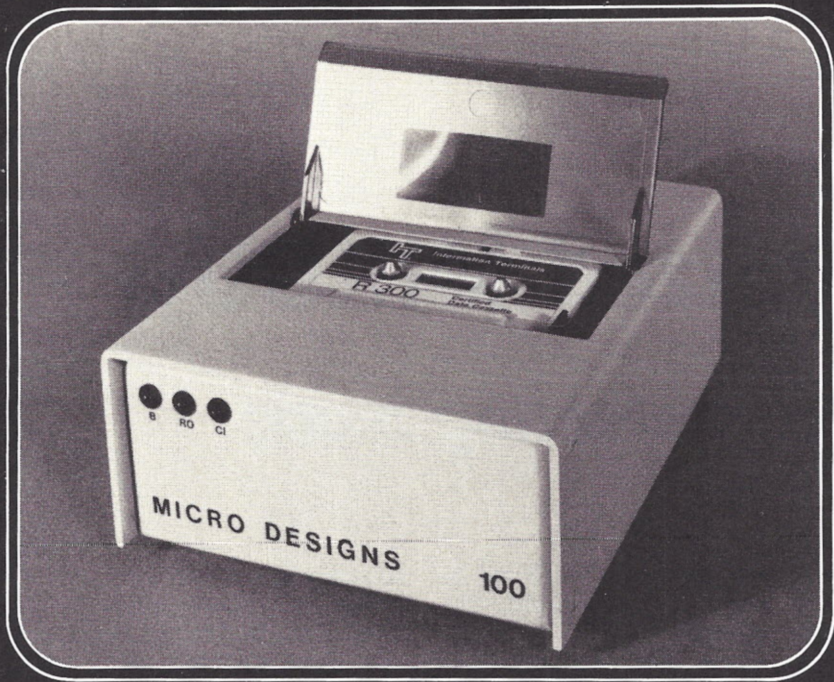
50 questions. The program prints out how many you got right and the total number of guesses. If you want to try again it goes to line 160, which sets certain variables equal to zero. The last section of the program is the data: the state followed by its capital.

This program was written for Imsai Disk BASIC and a 64-character-per-line video terminal. If you have hard copy only, you may want to skip some of the unnecessary printout or you will use up your paper supply. If you have a 32-column video terminal you should change the multiple-choice printout to one choice per line instead of two.

This program checks how well someone can match a state with its capital. By changing the printout wording and the data at the end, it could be used to match words with a short definition, a person with a certain event, etc. ■

```
YOU HAVE YOUR CHOICE OF FILL-IN OR MULTIPLE CHOICE
WOULD YOU LIKE TO FILL-IN THE ANSWERS? YES
FILL-IN — YOU MUST SPELL EXACTLY! (SAINT IS ABBREVIATED ST.)
TO STOP TYPE S FOR YOUR ANSWER

YOU HAVE YOUR CHOICE OF WHETHER THE STATE OR CAPITAL IS ASKED
WOULD YOU LIKE TO ANSWER WITH THE CAPITAL? NO

SALT LAKE CITY IS THE CAPITAL OF? UTAH
RIGHT! YOU HAVE 1 CORRECT

ATLANTA IS THE CAPITAL OF? GEORGIA
RIGHT! YOU HAVE 2 CORRECT

PHOENIX IS THE CAPITAL OF ? ARIZONA
RIGHT! YOU HAVE 3 CORRECT

OLYMPIA IS THE CAPITAL OF? S
DO YOU WANT TO STOP? YES

YOU GOT 3 RIGHT IN 3 GUESSES
WOULD YOU LIKE TO TRY AGAIN? YES

YOU HAVE YOUR CHOICE OF FILL-IN OR MULTIPLE CHOICE
WOULD YOU LIKE TO FILL-IN THE ANSWERS? NO
MULTIPLE CHOICE — ANSWER EACH QUESTION WITH 1, 2, 3 or 4
TO STOP TYPE 0 (ZERO) FOR YOUR ANSWER

YOU HAVE YOUR CHOICE OF WHETHER THE STATE OR CAPITAL IS ASKED
WOULD YOU LIKE TO ANSWER WITH THE CAPITAL? YES

1. HARRISBURG          3. MONTPELIER
2. BOISE               4. DENVER
THE CAPITAL OF VERMONT IS? 3
RIGHT! YOU HAVE 1 CORRECT

1. MADISON             3. CARSON CITY
2. MONTGOMERY          4. TOPEKA
THE CAPITAL OF ALABAMA IS? 1
WRONG

1. DOVER               3. LINCOLN
2. PIERRE              4. BISMARCK
THE CAPITAL OF NORTH DAKOTA IS? 5
DO YOU WANT TO STOP? NO
YOUR ANSWER FOR LAST QUESTION? 4
RIGHT! YOU HAVE 2 CORRECT
```

*Sample run.*




# THE FIRST REAL ALTERNATIVE TO DISK STORAGE FOR THE S-100 BUS

MICRO DESIGNS introduces the Model 100 and Model 200 Digital Cassette Mass Storage Systems. A high speed seek function and formatted data structure make these systems practical alternatives to floppy disks. These design features allow quick read and write access to individual records without altering other data on tape.

**Our systems offer:**

- Complete file management package on cassette with each unit.
- Random access capability — 4096 individually accessible records on each cassette.
- File copy capability on the dual drive Model 200.
- High data capacity — ½ megabyte per cassette.
- High speed seek to individual records at 120 in/sec. Worst case access less than 30 seconds.
- Hardware CRC for soft error recovery — Soft error rate less than 1 in $10^8$ bits.
- Bootstrap loader in onboard PROM.

The MICRO DESIGNS Models 100 and 200 are fully assembled and rigorously tested at the factory. They come ready for immediate use.

MODEL 100 . . . . $600
MODEL 200 . . . . $875

**Contact your local computer store for a demonstration or additional information.**

**Or write or call us directly.**

**MICRO DESIGNS, INC.**
**499 Embarcadero**
**Oakland, CA 94606**
**(415) 465-1861**

*Don Kinzer*
*3885 N.W. Columbia Ave.*
*Portland OR 97229*

# Customized MIKBUG

## *some interesting mods*

The majority of M6800-based hobbyist computer systems utilize the Motorola MCM6830L7 firmware system monitor, better known as MIKBUG. For all the things said about it, good and bad, it exists; for many of us it will have to serve, at least for the time being. While we are using it, however, we might as well take full advantage of its facilities and features. It is the intent of his article to suggest and explain several novel improvisations that make life a little easier and more interesting for those using a MIKBUG-based system.

### The PIA

It is assumed that everyone who reads this article has, or is familiar with, MIKBUG and its associated hardware, the MC6820 parallel interface adapter (PIA). Furthermore, it is assumed that the reader has access to *Engineering Note 100,* supplied with the MIKBUG ROM, which details its operation. Fig. 1 shows a circuit similar to the hardware suggested by Motorola to support MIKBUG. It is also similar to the MIKBUG-associated hardware on the Southwest Technical Products MP-C serial control interface board in their 6800 system.

The PIA (MC6820) consists of two nearly identical halves, each having eight lines that can be individually programmed as input or output data lines, as well as two control lines, one of which is input only and one that can be programmed as either an input or an output. These control lines are supported by full handshaking capabilities, including interrupt facilities. For a more complete description of the operation and programming of the PIA, consult the data sheet for the MC6820.

Fig. 1 shows that the A side of the PIA has six unused lines, as well as having both control lines free. The B side has four lines free and one unused control line. These free lines can be put to good use through clever programming and a little hardware help. Table 1 shows the characteristics of the lines as they are programmed by MIKBUG. The lines can always be reprogrammed (carefully), but the simplest improvisations are made with no changes.

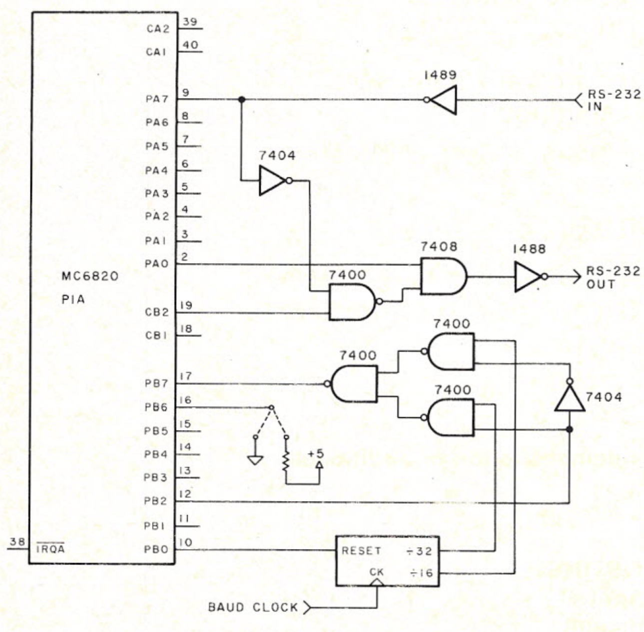



*Fig. 1. Typical MIKBUG harware support.*

### Sense Switches

The first operation we can perform on the PIA is to mount six SPST switches on the front panel of the system, with one terminal of each switch tied to ground and the remaining terminal tied to one of the six lines $PA_6$-$PA_1$. The result is six sense switches available for use as we wish. They are called sense switches because the computer can "sense" the state of the switch. Now, all we need is an application in which it is useful to know the condition of a switch. For future reference, the switches will be named SS6-SS1 and are connected to $PA_6$-$PA_1$, respectively. Program 1 illustrates how these sense switches can be used.

How many times have you been playing Star Trek, only to have the galaxy map roll off the top of your CRT before you can absorb all the information? You need a switch that can be thrown to halt further output until it is returned to its previous position. A perfect application for a sense switch! Let's use SS6 and the new output character shown in Program 1.

If SS6 is closed, the routine "hangs" in a loop until the switch is opened again, at which point it proceeds to print the character. Now replace every subroutine call to $E1D1 with a call to $A014. Neat and simple. The routine is even relocatable!

Let's assume you have a printer, such as the SWTPC PR-40, on which you want to control the printout; but you want full-time printout on your CRT terminal. Solution: Call the routine in Program 2 that uses SS5.

If SS5 is closed, the characters are printed on the CRT only. If open, they are printed on the printer and CRT. Note that the address of your printer driver routine should be placed in locations $A055 and $A056.

Perhaps you have a KC standard audio cassette interface (the SWTPC AC-30, for example) connected in parallel with your CRT. You want to record a leader and a title on the tape before the program. Simple. Use Program 3, which senses the position of SS4.

To use this routine, simply set up the punch limits in MIKBUG ($A002, $A004), close SS4, and then GO to this program. The CPU will execute the loop consisting of the first three lines until SS4 is opened. Meanwhile, the output line to the cassette interface will be marking, generating your leader. To add a title, first type L, and then a few spaces. The MIKBUG hardware echo will send the characters back to the cassette interface as you

type them. When you're ready to record the program simply open SS4.

There are many more applications for sense switches. I hope these examples will stimulate your imagination. For applications in which mechanical switch bounce is intolerable, the sense switches must be debounced. A standard circuit for hardware debouncing using an SPDT switch and two 7400 NAND gates is shown in Fig. 2.

The switches can also be software debounced. Program 4 illustrates this function. The routine is written for SS3 (indicated by the mask constant $08) and works by sampling the switch, delaying and comparing the switch state. If the state is the same, everything is fine; otherwise the switch is sampled again.

**Program Interrupt Switch**

The next improvisation on MIKBUG could be applied to almost any computer system. Many times in the course of programming and debugging, it is helpful to stop the CPU in such a manner that execution can begin at the point where it stopped. If you know where you want it to stop, a software interrupt (SWI), or breakpoint, is the solution. However, if the machine is executing a program and you want to stop it to see what it is doing, you can't. That is, you can't unless you have connected a switch to the system as shown in Fig. 3. When the switch is closed, the CPU will be interrupted (assuming the interrupt mask is cleared). This process in the 6800 saves the current state of the machine for future use. At this point, we take advantage of the software interrupt handler in MIKBUG. The interrupt service routine for the Program Interrupt (PI) is shown in Example 1. The IRQ vector ($A000) should be set in this case to A04A (i.e., point to the service routine). Now when the switch is closed, the contents of the registers will be printed as if we had hit an R while in MIKBUG. Furthermore, typing G will resume execution where it stopped. It is necessary to set up the stack pointer (SP) in the program being executed using the following instruction.

```
8E A0 7F     LDS     #$A07F
```

This prevents your program

```
A014 36          OUTCH    PSH A                  SAVE CHARACTER
A015 B6 80 04    DISABL   LDA A      PIADR       GET SWITCHES
A018 84 40                AND A      #$40        MASK OUT # 6
A01A 27 F9                BEQ        DISABL      IF ON, LOOP
A01C 32                   PUL A                  GET CHARACTER BACK
A01D 7E E1 D1             JMP        $E1D1       GO PRINT IT
```

*Program 1.*

```
A04A 36          PRINT    PSH A                  SAVE CHARACTER
A04B B6 80 04             LDA A      PIADRA      GET SWITCHES
A04E 84 20                AND A      #$20        MASK OUT #5
A050 27 05                BEQ        NOPR40      IF ON, SKIP
A052 32                   PUL A                  GET CHARACTER
A053 36                   PSH A                  PUT IT BACK
A054 BDXX XX              JSR        PRTCH       SEND TO PR-40
A057 32          NOPR40   PUL A                  GET CHARACTER
A058 7E E1 D1             JMP        $E1D1       GO PRINT ON CRT
```

*Program 2.*

```
A014 B6 80 04    LEAD     LDA A      $8004       GET SWITCHES
A017 84 10                AND A      #$10        MASK #4
A019 27 F9                BEQ        LEAD        IF OFF, LOOP
A01B 7E E1 3D             JMP        $E13D       GO PUNCH
```

*Program 3.*

```
A014 B6 80 04    DEBNCE   LDA A      $8004       GET SWITCHES
A017 16          LOOP     TAB                    SAVE
A018 CE 40 00             LDX        #$4000      SET DELAY VALUE
A01B 09          DELAY    DEX                    COUNT DOWN
A01C 26 FD                BNE        DELAY       IF NOT DONE, LOOP
A01E F8 80 04             EOR B      $8004       COMPARE SWITCH STATE
A021 C4 08                AND B      #$08        MASK #3
A023 26 F2                BNE        LOOP        IF NOT SAME, DO AGAIN
A025 84 08                AND A      #$08        MASK OUT #3
A027 39                   RTS                    DONE
```

*Program 4.*

```
A04A BF A0 08    STS    $A008    SAVE SP
A04D 7E E1 1F    JMP    $E11F    GO TO REGISTER DISPLAY ROUTINE
```

*Example 1.*

| Line | Function |
|---|---|
| $PA_6$-$PA_1$ | input |
| $PB_5$-$PB_3$ | input |
| $PB_1$ | output |
| CA1 | interrupt input, rising edge sensitive. |
| CA2 | non-interrupt input, falling edge sensitive. |
| CB1 | non-interrupt input, rising edge sensitive. |

*Table 1. Unused PIA line functions as programmed by MIKBUG.*

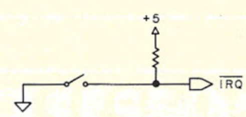



*Fig. 3. Program interrupt switch.*

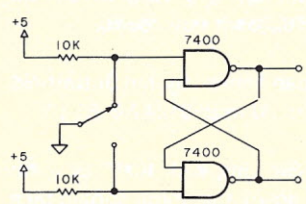



*Fig. 2. Switch debouncing circuit.*

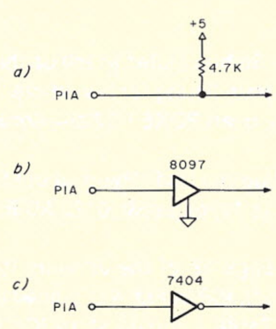



*Fig. 4. PIA line protection.*

```
BDE1 AC        INTERR     JSR      INCH     GET CHARACTER
97 20                     STA  A   SAVE     STASH IT IN BUFFER
3B                        RTI               RETURN
```

*Program 5.*

```
86 0F          INIT       LDA  A   #$0F
B7 80 05                  STA  A   $8005
```

*Example 2.*

| Register | Address |
|---|---|
| Peripheral Register A | $8004 |
| Control Register A | $8005 |
| Peripheral Register B | $8006 |
| Control Register B | $8007 |

*Table 2. MIKBUG PIA register addresses.*

stack and MIKBUG's stack from interfering with one another during the interrupt. It is also recommended that the interrupt mask be set (SEI) before doing I/O and cleared (CLI) afterward. This prevents the characters from being interrupted during transmission.

**Reprogramming PIA Lines**

The PB1 line of the PIA is programmed by MIKBUG to be an output, and there are many applications for a logic level control line. Some suggestions are process control lines, blinking lights and music generation. Since the user can reprogram the unused PIA lines to suit current needs, up to 13 control lines (including PIA control lines) can be utilized without the user having to add more hardware. Note: It is advisable to protect the PIA lines using one of the means that is shown in Fig. 4.

This helps prevent destruction of the PIA due to static charge buildup, and the second and third solutions also increase the drive capability of the control line. The third solution, of course, inverts the control line. Table 2 should be of help to those who intend to reprogram MIKBUG PIA lines.

By reprogramming, we can realize any number of functions. For example, by adding a few jumpers and a little software, we can have interrupt-driven input from the CRT. To realize this goal, it is necessary to connect CA2 (pin 39) to PA7 (pin 9) on the MIKBUG PIA and also connect IRQA (pin 38) of the PIA to the IRQ line of the bus. With these jumpers connected, MIKBUG will not operate properly unless the interrupt mask is set as it is upon power-up or after pressing the reset button. Since this is the case, you might want to make the jumper from pin 9 to pin 39 through a switch allowing quick disabling of the interrupt mode. The trick to the software is in the reprogramming of the PIA. We need to detect the start bit of an incoming character, so it is necessary to program CA2 to interrupt on the falling edge. This is accomplished by the sequence in Example 2.

An interrupt routine to service the interrupt requests generated by an incoming character is shown in Program 5. The routine assumes a one-character buffer at location $0020 and that the CRT is the only interrupt source. More complex buffer arrangements can be utilized by modifying the code. Furthermore, character processing, e.g. checking for special characters, etc., can be done in the service routine. Note that if the interrupt mask is set, INCH will operate normally, yielding normal program-driven input. However, when the mask is cleared, interrupt-driven input is enabled. Note: This scheme will not work at speeds above 1200 baud because of MIKBUG's critical timing.

The ideas presented in this article represent only a few of many possibilities. I hope they will be useful. Further improvisations are limited only by your cleverness and imagination. ■

**References:**

*Engineering Note 100,* Motorola Semiconductor Products, Inc.

*M6800 Microprocessor Applications Manual,* Motorola Semiconductor Products, Inc.

*The TTL Data Book for Design Engineers,* Texas Instruments, Inc.

*M6800 Microcomputer System Design Data,* Motorola Semiconductor Products, Inc.

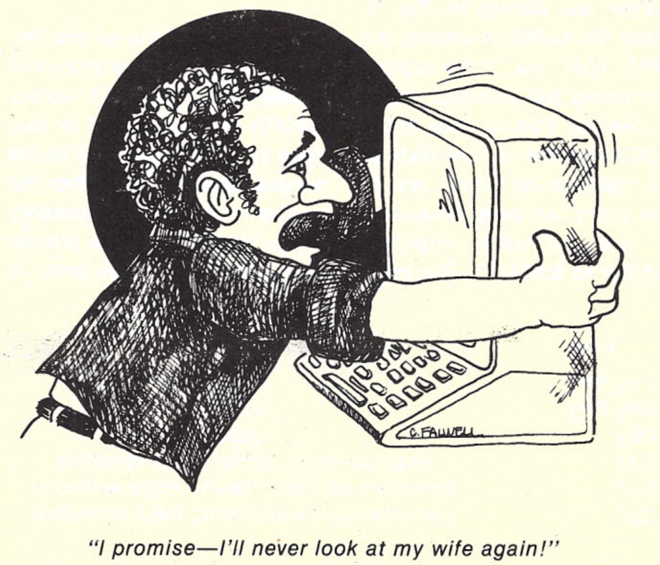
*M6800 Programming Reference Manual,* Motorola Semiconductor Products, Inc.

# CORRECTIONS

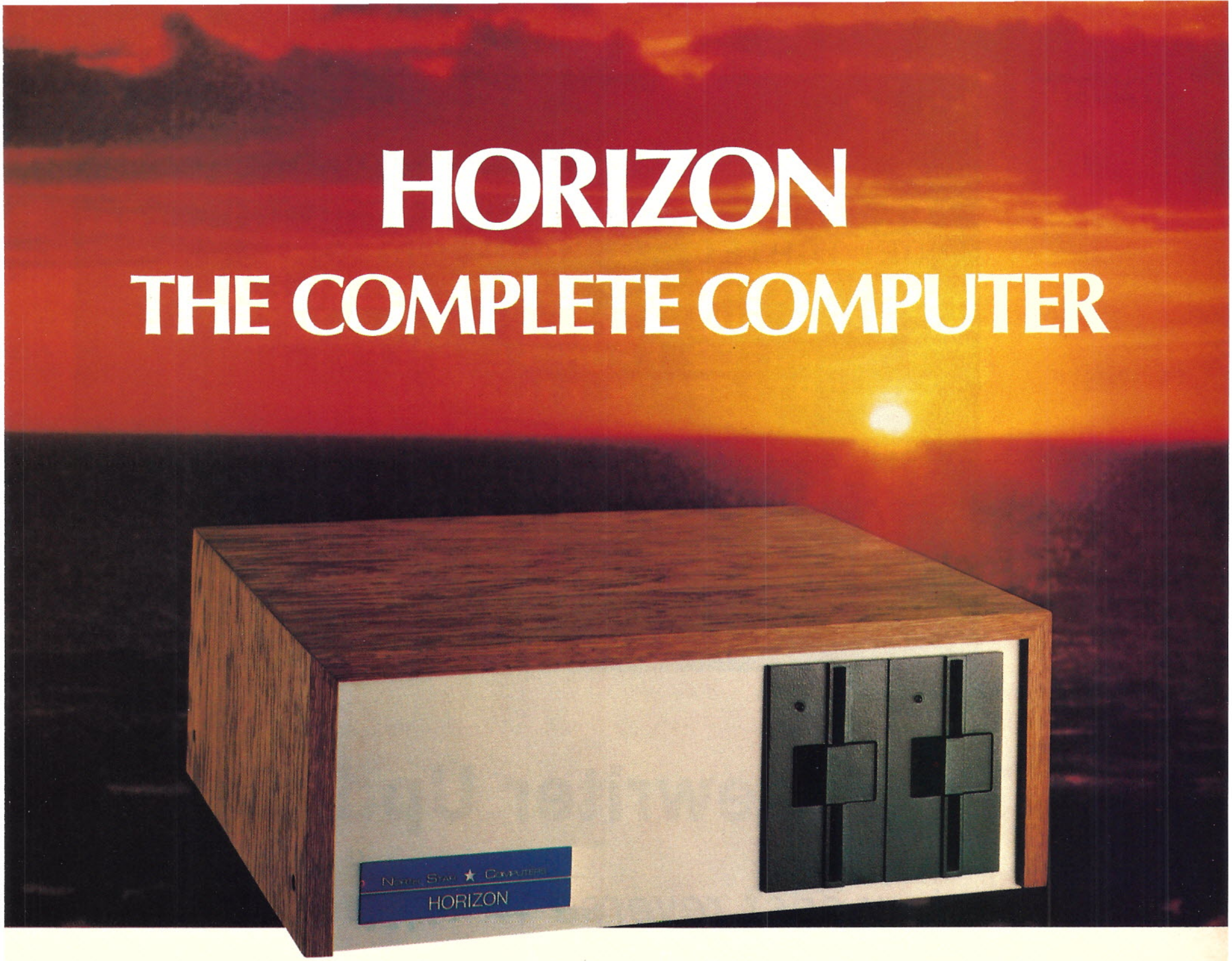
Bob Bishop called to tell us that the last element of line 900 in the program listing for his article "Rocket Pilot" (No. 13, p. 91) should have been POKE 50,225—not POKE 50,2 as it now reads.

In Program D of "Hyper about Slow Load Times" by Jim Butterfield (issue 11, page 68), 0123 AD F5 A7 should read 0123 AD F5 17.

On page 39 of the January issue ("Growing with KIM" by John Eaton), IC8 in Fig. 4 is placed upside down; the notch should face up. Table 1 should show ICs 1 and 2 as 8833, and IC 6 as 74145 BDC/dec decoder.




*"I promise—I'll never look at my wife again!"*




# Look To The North Star HORIZON Computer.

**HORIZON™**— a complete, high-performance microprocessor system with integrated floppy disk memory. HORIZON is attractive, professionally engineered, and ideal for business, educational and personal applications.

To begin programming in extended BASIC, merely add a CRT or hard-copy terminal. HORIZON-1 includes a Z80A processor, 16K RAM, minifloppy™ disk and 12-slot S-100 motherboard with serial terminal interface — all standard equipment.

**WHAT ABOUT PERFORMANCE?**

The Z80A processor operates at 4MHZ — double the power of the 8080. And our 16K RAM board lets the Z80A *execute at full speed.* HORIZON can load or save a 10K byte disk program in less than 2 seconds. Each diskette can store 90K bytes.

**AND SOFTWARE, TOO**

HORIZON includes the North Star Disk Operating System and full extended BASIC on diskette ready at power-on. Our BASIC, now in widespread use, has everything desired in a BASIC, including sequential and random disk files, formatted output, a powerful line editor, strings, machine language CALL and more.

**EXPAND YOUR HORIZON**

Also available—Hardware floating point board (FPB); additional 16K memory boards with parity option. Add a second disk drive and you have HORIZON-2. Economical serial and parallel I/O ports may be installed on the motherboard. Many widely available S-100 bus peripheral boards can be added to HORIZON.

**QUALITY AT THE RIGHT PRICE**

HORIZON processor board, RAM, FPB and MICRO DISK SYSTEM can be bought separately for either Z80 or 8080 S-100 bus systems.

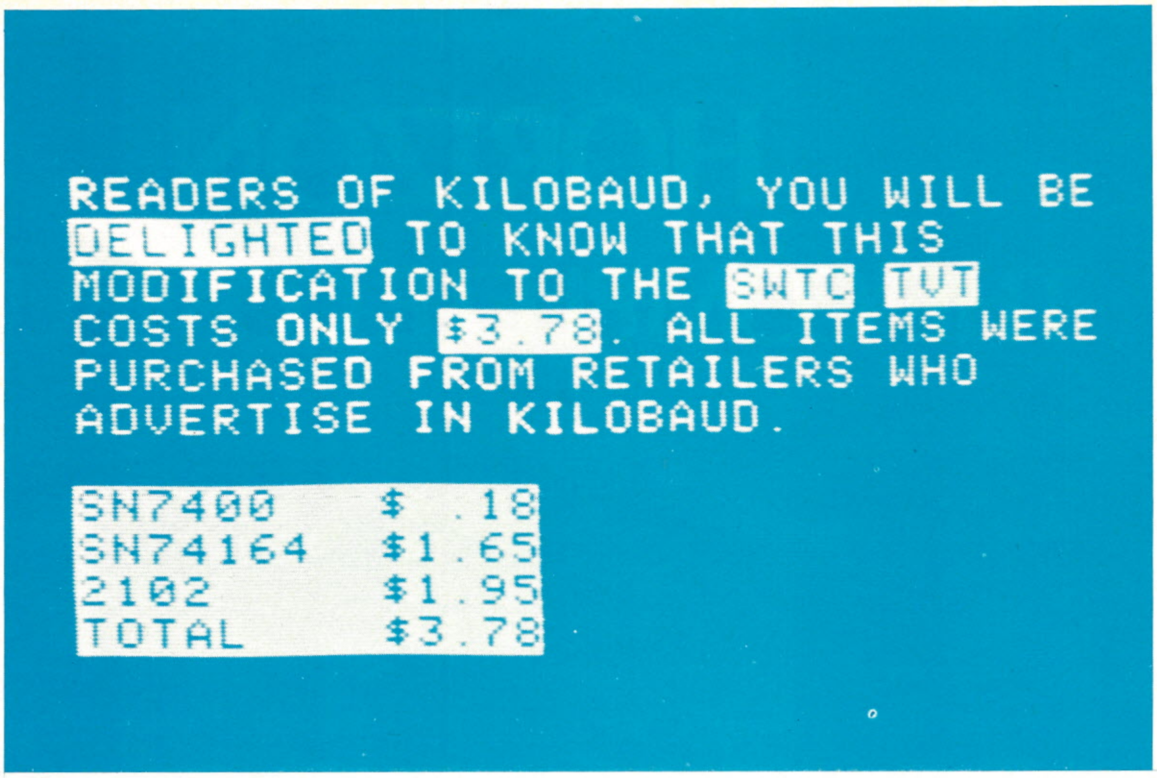
HORIZON-1 $1599 kit; $1899 assembled.
HORIZON-2 $1999 kit; $2349 assembled.

16K RAM—$399 kit; $459 assembled; Parity option $39 kit; $59 assembled. FPB $259 kit; $359 assembled. Z80 board $199 kit; $259 assembled. Prices subject to change. HORIZON offered in choice of wood or blue metal cover at no extra charge.

Write for free color catalogue or visit your local computer store.

NORTH STAR ★ COMPUTERS N9
2547 Ninth Street • Berkeley, California 94710 • (415) 549-0858

*Clint H. Woeltjen*
*La Habra CA*




# TV Typewriter Update

## *low-cost reverse video mod*

When I first got my SWTPC TVT, I was thrilled to be able to type characters on a TV screen. It didn't occur to me to want anything more. Later, I saw other TV terminals that permitted reversed images (characters or whole pages), so I decided to see what I could do with my own TVT. Inverting the signal from the output shift registers was simple enough, producing a reversed image for the whole page, but I wanted to be able to reverse individual characters. After a bit of experimenting, I was able to do just that.

### How It Works

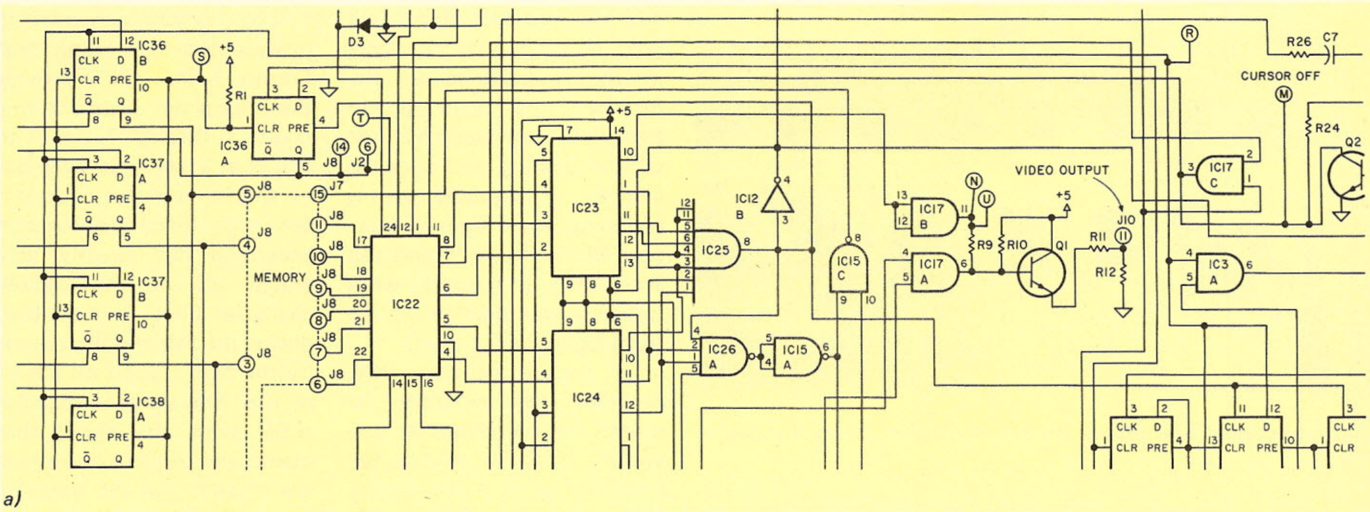
Point N (see Table 1 and Fig. 1a) is the output of shift registers IC23 and IC24 (buffered through IC17B) and contains the character data from the character generator IC22 (also see Fig. 1a). By inverting the signal at this point, you can produce a negative image. (Black letters on a white background, rather than the normal white on black.) The circuits formed by ICA and IC15D, shown in Fig. 2, produce inverted data at point U, the input to the video output whenever the output (pin 11) of ICB is logic 1. The data is presented at point U in uninverted form whenever pin 11 of ICB is logic 0.

The memory IC (ICC) stores the background data and is addressed by the same circuitry that addresses the six bits of the main memory (see Fig. 3). The latch in Fig.

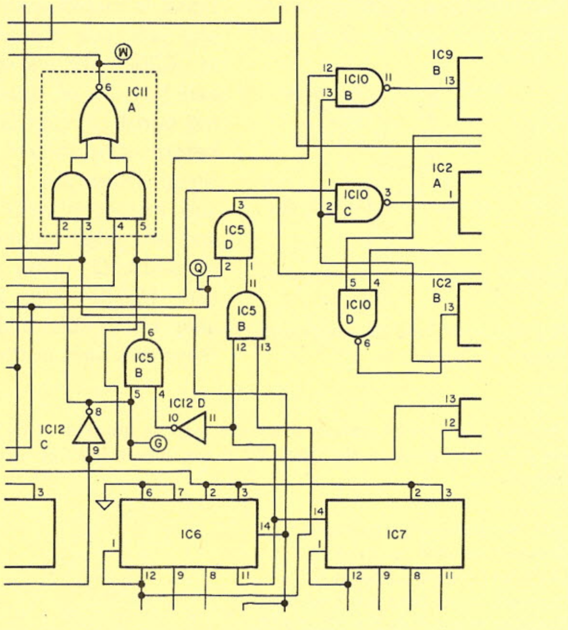
Point A is pin 1 of IC28.
Point B is pin 2 of IC28.
Point C is pin 3 of IC28.
Point D is pin 11 of IC28.
Point E is pin 12 of IC28.
Point F is pin 13 of IC28.
Point G is the line connected to pin 8 of IC12.
Point H is pin 1 of IC4
Point I is pin 2 of IC4.
Point J is pin 3 of IC4.
Point K is pin 4 of IC4.
Point L is pin 5 of IC4.
Point W is the line connected to pin 6 of IC11.
Point N is the line connected to pin 11 of IC17.
Point O is pins 1 and 2 of IC15 shorted together.
Point P is pin 3 of IC15.
Point Q is the line connected to pin 13 of IC19.
Point R is the line connected to pin 5 of IC9.
Point S is the line connected to pin 1 of IC36.
Point T is the line connected to pin 5 of IC36.
Point U is the end of R9 previously connected at point N.

Note: When the point referred to is an interconnecting line, there is a plated-through hole on that line that can be used for that point. Follow the plated line on the circuit board from the pin mentioned until you find the plated-through hole, attach a wire to that hole and label it with the proper point designation.

*Table 1. Verbal schematic of modifications required for image reversal.*

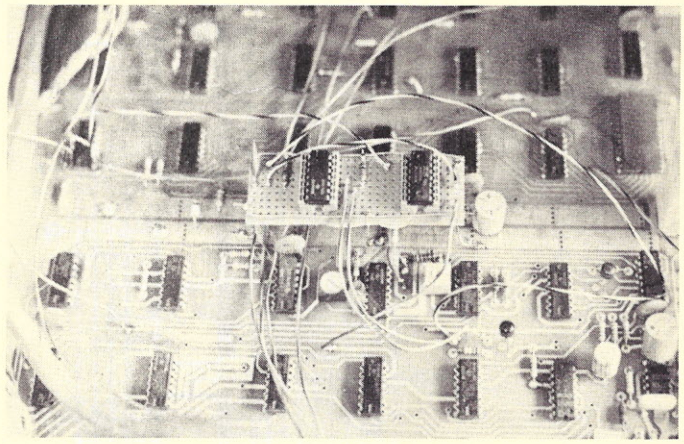



a)




b)

Figs. 1a & 1b. Take-off points for the modifications of the main board.



*Background decoding and sync circuit board described in Fig. 2 showing ICA and ICB prior to final wiring. Shown mounted over B+ and ground buses for added mechanical stability.*

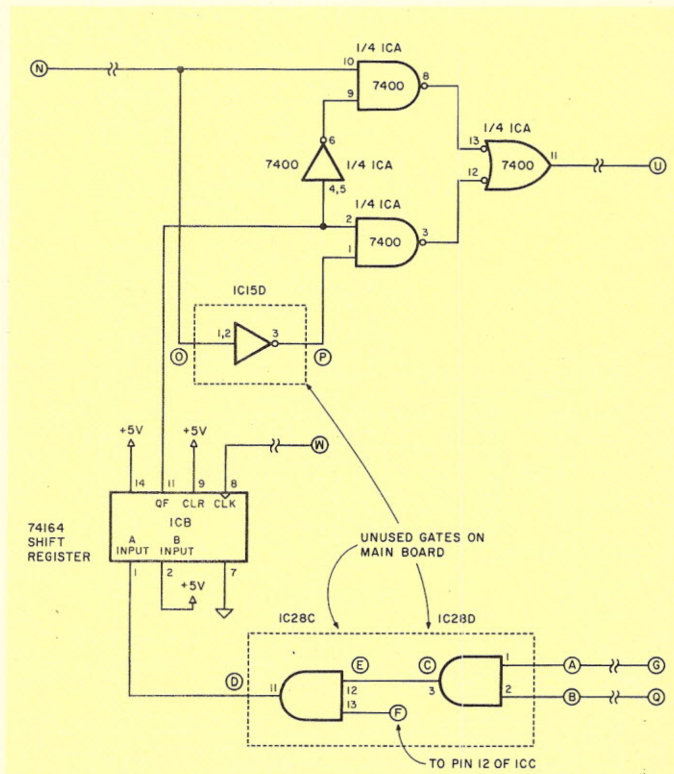
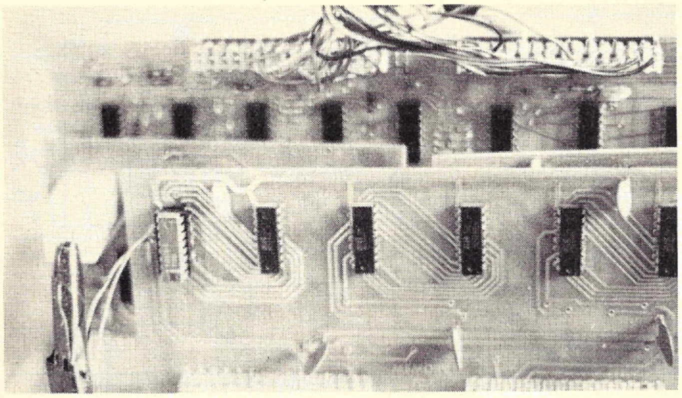



*Fig. 2. Background decoding and sync circuit.*

4, IC4A, is used for temporary storage of the background data and is paralleled with ICs 36-39 to provide synchronized operation. Unfortunatley, the character generator ROM is slower than ICs 28C and 28D, so when the data for the image and the background are mixed, they are out of sync. (The background appears to be shifted five or six "dots" to the left of the character.) This is remedied by the addition of the shift register in Fig. 2, ICB, which is clocked by the dot clock (point W, Fig. 1b). With the SN74164 shift register, you have the choice of shifting from one to eight bits. Taking the output at bit 6 provides for the delay required to bring the background back into sync with the character.

A full frame of a TV picture consists of 264 lines (at least as far as the SWTPC TVT goes), but the character data uses only 160 lines. The remaining 104 lines are counted by the same counter as the character address lines, so if we don't use some sort of blanking, the first 104 characters will be repeated along the top and bottom borders of the screen. The TVT has blanking circuitry that prevents this, and provides the borders and vertical spacing between characters. Using the existing blanking circuitry provides you with the desired blanked border,



*ICC piggybacked over IC6 on the memory board. Circuit is shown in Fig. 3.*

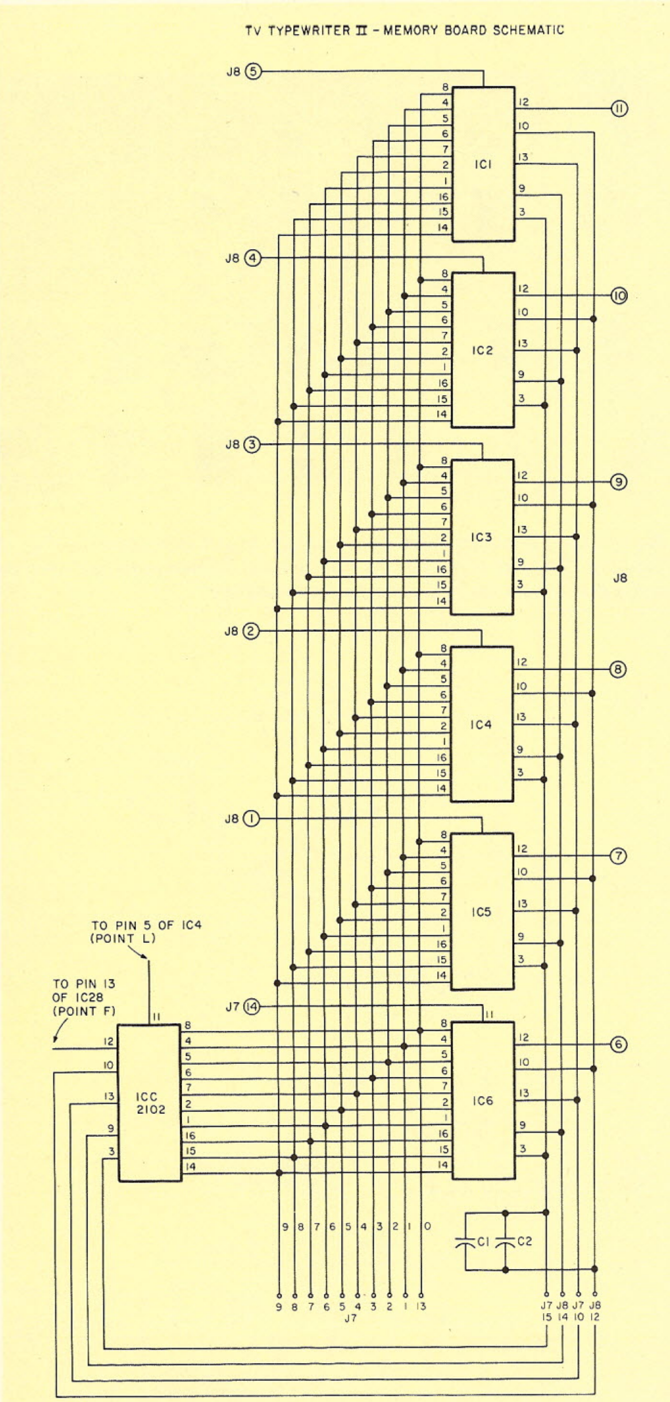



*Fig. 3. Memory board modification for the image-reversal bit.*

but also provides two black lines beneath each character in the reversed mode. Remember, the blanking circuitry is provided for the vertical spacing, and since the inverter is disabled during blanking, these lines will appear black. I didn't find this acceptable, so I set to work isolating the portion of the blanking circuit at fault.

A decade counter, IC6, provides the circuitry required for the character generator, IC22, to decode the lines in the 5x7 dot matrix that makes up each character. The initial count, 0, is used by the character generator to provide the first blank line. Counts 1 through 7 provide the character lines. During counts 0 through 7, pin 11 of IC6 is at a low logic level, but on the 8th and 9th counts, it goes high. Since the character generator has already generated all of the desired character lines, it is disabled during the 8th and 9th counts by the high output on pin 11 of IC6. This signal is mixed at IC5B with the signal that blanks the top and bottom of the screen, and the resulting signal is mixed with the right and left side blanking signal at IC29D. By taking the top and bottom of screen blanking signal at point G (Fig. 1b) before it is mixed with the signal from pin 11 of IC6, and mixing it with the right and left edge blanking signal at point Q, using the AND gate IC28D (Fig. 2), I was able to inhibit blanking on these two lines in the reversed

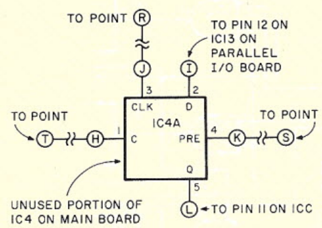



*Fig. 4. Latch used for holding the image-reversal bit.*

Remove the lead of resistor R9 from land at point N. This lead forms point U. The land on the main PC board where R9 was located is now point N.

Connect point A to point G.
Connect point B to point Q.
Connect point C to point E.
Connect point D to pin 1 of ICB.
Connect point F to pin 12 of ICC.
Connect point H to point T.
Connect point I to pin 12 of IC13 on parallel I/O board.*
Connect point J to point R.
Connect point K to point S.
Connect point L to pin 11 of ICC.
Connect point W to pin 8 of ICB.
Connect point N to point O and pin 10 of ICA.
Connect point P to pin 1 of ICA.
Connect point U to pin 11 of ICA.

The following changes are on the parallel I/0 board.

Connect pin 9 of IC5 to pin 5 of IC5.
Connect pin 10 of IC5 to bit 8 from computer.
Connect pin 8 of IC5 to pin 14 of IC13.
Connect pin 13 of IC13 to center off SPDT switch (see Fig. 5).
Connect pin 12 of IC13 to point I on main PC board.

*If you don't have a parallel I/O board, or don't want computer control, point I can be attached to the center of a SPDT switch, with one leg going to a 1k resistor to +5 volts and the other to ground. If you want both computer control and manual operation, use a center off switch. I don't have a SWTPC serial I/O board, so I can't tell you how to modify it for this function, but it shouldn't be too much of a problem.

I soldered a 16-pin IC socket with pins 11 and 12 bent outward directly to IC6 on the memory board. Don't use more than a 15 Watt iron, and be very careful. Solder just long enough to ensure connection. ICC will plug directly into this socket.

*Table 2. Construction outline for modifying the TVT for image reversal.*

mode. Now I have full use of the image reversal.

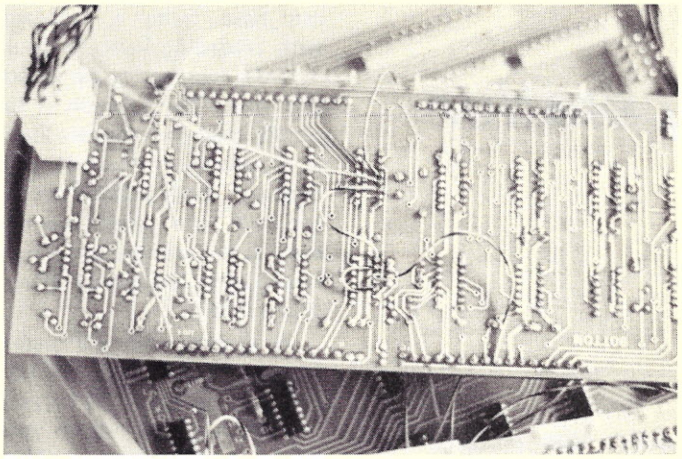
### Construction

I wire-wrapped the circuit in Fig. 2 and mounted it with 16-gauge wire above the 5 V and ground buses on the main board. This is a central location, and this technique provides sufficient mechanical stability.

To provide computer control of the background, I modified my parallel I/O board as shown in Fig. 5. I connected pin 9 of IC5 on the parallel I/O board to the true/inverted bus (the line that pins 2, 5 and 12 of IC5 are connected to). Pin 10 is connected to bit 8 of the data byte from your computer. Pin 8 is connected to pin 14 of IC13. IC5 is a quad Exclusive OR gate, with one gate unused by the original circuit. IC13 is a quad 1 of 2 data selector with one circuit unused. Pin 13 of IC13 can be switched by a center off SPDT switch, with one side grounded and the other side connected to +5 V through a 1k resistor. Pin 12 of IC13 goes to point I on the main board.

### Summary

As you can see, it isn't that hard to modify the SWTPC TVT to make it behave like a more complex terminal. Just three extra ICs, a little solder and a bit of time, and you've worked a miracle. I have provided a construction outline in Table 2 to help you work your own miracles with your TVT.■



*Modifications to the parallel I/O board to allow for computer control over the image-reversal bit. Circuit is shown in Fig. 5.*

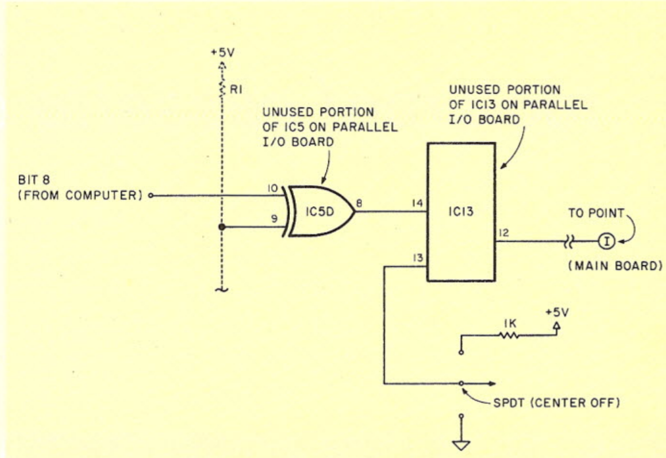



*Fig. 5. Modifications on the parallel I/O board to permit computer control of the image-reversal bit.*

*Stephen Gibson*
*PO Box 38386*
*Los Angeles CA 90038*

# Hear It and See It!

## *foolproof cassette operation*


*The Data Duffer.*

How many times have you dumped data out to cassette, only to discover (much later of course) that you had forgotten to plug in the right audio cable to your recorder? Or how about the time you tried to load BASIC and you found you had forgotten to reset the counter on the recorder, only to start the feed right smack dab in the middle of your data? So you go back to the beginning. But this time you've forgotten to adjust the volume level. Exasperating isn't it?

If you own a Mits ACR, you well know that loading 8K BASIC begins to feel like 4 hours rather than 4 minutes. Even if you have a Tarbell high speed interface or one of the others, you well know the aggravation not to mention the embarrassment during a demonstration! You mutter, "Hmm . . . data didn't load properly." The wide-eyed onlookers of course ask why. "Well it must be a discombobulated transfer function!" as you plug in the right cable hoping no one noticed.

### This Nonsense Has Got To Stop

The Data Duffer is a box of solutions to the above problems . . . and you don't have to modify your recorder to the hilt. The only problem with this little jewel is simply "Where do you put it?" If you're like me, you've got more than enough wires and peripheral goodies cluttering your installation. I put Data Duffer UNDER my cassette recorder, as shown in the photo. Some neat stuff called Velcro keeps your recorder from falling off. The controls are very handy in this position.

### Super Simple Circuit

Fig. 1 is schematic diagrams of the circuits involved. The cassette I/O cables are permanently plugged into your recorder, thereby eliminating the problem of leaving a cable out, or even worse, reversing them! The I/O switch S1 (the knob on the front panel) controls send and receive. It solves the problem some interfaces have with the simultaneous reading and writing of data. The switch also lifts the opposite cable shield above ground. This is very important, since some recorders have the low side of their seemingly unbalanced output jacks isolated above ground by several hundred Ohms. You can have quite a problem with recording levels if your recorder happens to be wired this way. The solution is to simply treat the recorder lines as though they were balanced lines (i.e., both isolated above ground), and then ground only at the computer's interface device. Switch S1 solves this problem.

A speaker and VU meter are wired across the OUTPUT jack line ahead of S1 so you can *see and hear* where you are in a data transfer. This arrangement is very useful to those with recorders whose monitor jacks are connected to the same driver amplifier for the record/play head. Not only do you know the volume level, but you know you really *are* writing out to the cassette.

The speaker loaded my recorder's monitor line too much. So I added a simple LM380 IC amplifier. This Amp has got to be the simplest IC Amp around. Power is stolen from the +16 to 18 V supply in the computer. You may not need the Amp, but it sure would come in handy when you get around to playing music or giving your computer a voice.

The LED is optional but useful to those with Mits ACRs or Tarbell interfaces where data phase is important. It's merely wired in place of the one hidden deep in the computer (on the interface board) and brought out to the Data Duffer's front panel.

### Construction Tips

The chassis layout and parts placement are shown in

the photo. R2, a PC board potentiometer, is mounted on the meter. Cable ties are used to bundle the lines in the box as well as those leading to the computer. They terminate in a conventional Cannon DB-25P connector, sometimes called an RS232 connector.

I mounted the speaker in the bottom of the box. Interestingly enough, the size of the box coupled to the speaker (5" x 7") produce a very interesting sound. The amplifier is mounted on a Radio Shack ready-made PC board for one DIP socket. The board is then bolted to the cabinet back with 6-32 hardware to provide a measure of heat sinking which is not really necessary . . . it's just inbred in me from a few years of equipment building. (Refer to Fig. 2.) The speaker control, R1, is mounted on the back because you probably won't use it very much.

The I/O lines for the recorder are brought up through the top of the box and plug into the recorder. Use grommets in the holes. Drill a hole above the meter and insert a grommet. You can then press-fit an LED into the hole. The black ring of the grommet provides better contrast to see the LED in high ambient light.

Velcro, a fabric fastener, may be super-glued to the top of the box and to the bottom of the recorder to keep the recorder in place. You can get this stuff at nearly any yardage shop. Use plenty of super-glue to stick it all together and you'll have a solid holder for the recorder, but at the same time, you can lift it off to record that next computer club guest speaker.

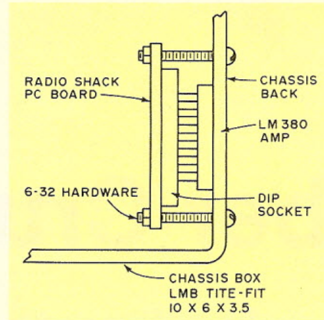
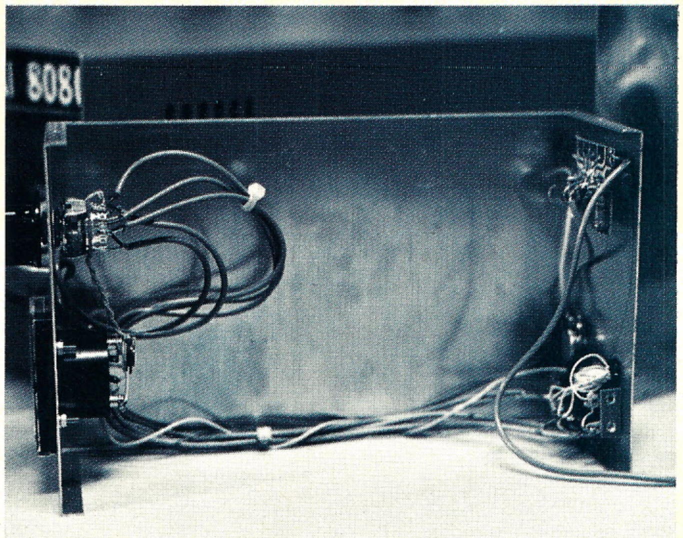



*Fig. 2. Detail of simple LM 380 amplifier mounting technique for "died in the wool" heat sink fans.*



*Speaker is mounted to bottom of box. LM 380 Amp and volume control R1 mount on rear.*

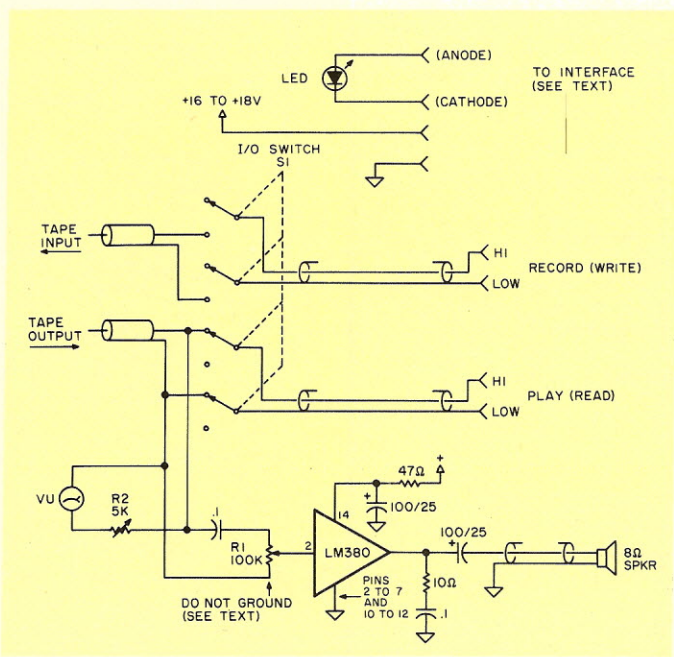



*Fig. 1. Simple circuit solves grounding problems with computer. LM 380 Amp and VU meter allow you to "see and hear" data transfer.*

As a worthwhile final touch, apply some press-stick letters (available at stationery stores) to decorate the front panel. You may not think it's worth the bother, but I can tell you that it's easier than it looks. Why not do it? A shot of clear acrylic spray is all that's needed to keep the letters from being scratched.

## ACR LED Modification

"But my ACR doesn't have one of those neat LEDs," you say. Never fear because Fig. 3 shows you how to add one. Simply remove the jumper from A to K. Then connect the LED anode to A and the Cathode to K . . . voila! Or you could run the wires from A and K out to the LED on the Data Duffer. You choose where you want it. Just be sure the LED polarity is correct.

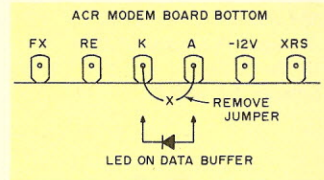



*Fig. 3. Mits ACR modified to detect carrier by adding LED in place of jumper A to K.*

## Don't Lose Those Bits

You may have heard this before, but not all cassette tape is the same. Here is a list of various tapes I've tried which give satisfactory results.

Radio Shack Super Tape
Maxell UD
Sony Low Noise
BASF Low Noise
Scotch Hilander Low Noise

Bias levels in your recorder may favor one tape over another. Once you've found the brand for your machine, stick with it. You're asking for trouble with the cheap stuff.

## Calibration . . . a cinch

To calibrate the meter, 1) feed in a carrier or sync stream. If you haven't a test tape of either . . . try data! 2) Adjust your recorder volume to set the best level. Use the LEDs mentioned above or better yet a scope at the appropriate point in your interface. 3) When you find a midrange setting, adjust R2 for a "0" VU reading on the meter. That's all. You now have a valid indicator of what's going in as well as out of your computer.

## Why Build This Thing?

I suspect that the Data Duffer looks simple enough to build that many hardware types won't even consider it. "After all, Steve, the outputs aren't buffered!" But beware! If you've ever had data I/O problems such as those I've described, you really *need* this device. Besides, the *data sound* is a cheap bells-and-whistles effect to add a little showmanship to your installation, not to mention the blessèd relief a solid I/O transfer brings *you!* ■

*Mel Baker*
*Assistant Professor of Math*
*Union College*
*Lincoln NE 68506*

# Number-Crunching Time

## *algebra the easy way*

How does a beginning algebra student solve the equation $x + 7 = 9$? Probably by trial and error. Since $2 + 7 = 9$ he concludes that $x = 2$. His teacher will likely insist that he subtract 7 from both sides despite the seeming foolishness. The teacher insists that such techniques be used in anticipation of more difficult equations such as $\frac{x}{3} = 2x + 9$. When the student considers an equation like $\sin(x + 2) - e^x + x^2 + 3.701 = 0$, his math teacher may be at a loss to suggest a method of solution.

Let us call the left member of this equation y. The equation then takes the form y = 0. All equations can be written in the form y = 0. For example $\frac{x}{3} = 2x + 9$ may be written $\frac{x}{3} - 2x - 9 = 0$.

With the computer, we can return to the trial and error method suggested in the beginning.

For example, if x is given the value 1 in $\sin(x + 2) - e^x + x^2 + 3.701$, y computes to 2.12384. When x is given the value 2, y computes to -.444859. Since one value of y is positive and the other negative, we suspect that y =

```
INITIAL VALUE OF X AND STEP SIZE ARE? -11,1

X          Y

-11        124.289
-10        102.712
-9         84.0439        Search for unit interval.
-8         67.9801
-7         53.659         14 iterations.
-6         40.4553
-5         28.5531
-4         18.7734
-3         11.8097
-2         7.56566
-1         5.17459
0          3.6103
1          2.12384
2          -.444859
-------------------------------------------------
1.1        1.94841
1.2        1.76251
1.3        1.56396        Search for 0.1 interval.
1.4        1.35026
1.5        1.11853        9 iterations.
1.6        .865447
1.7        .587216
1.8        .279495
1.9        -6.26606E-2
-------------------------------------------------
1.81       .246916
1.82       .213988        Search for 0.01 interval.
1.83       .180705
1.84       .147063        9 iterations.
1.85       .113055
1.86       7.86768E-2
1.87       4.39216E-2
1.88       8.78421E-3
1.89       -2.67413E-2
-------------------------------------------------
1.881      5.24922E-3
1.882      1.71034E-3     Search for 0.001 interval.
1.883      -1.83242E-3    3 iterations.
-------------------------------------------------
1.8821     1.35624E-3
1.8822     1.00210E-3     Search for 0.001 interval.
1.8823     6.47923E-4
1.8824     2.93705E-4     5 iterations.
1.8825     -6.05519E-5
-------------------------------------------------
1.88241    2.58281E-4
1.88242    2.22857E-4
1.88243    1.87432E-4     Search for 0.00001 interval.
1.88244    1.52007E-4
1.88245    1.16582E-4     9 iterations.
1.88246    8.11556E-5
1.88247    4.57293E-5
1.88248    1.03026E-5
1.88249    -2.51244E-5
-------------------------------------------------
1.88248    6.75995E-6     These 3 iterations are redundant in this
1.88248    3.21725E-6     problem as 6 significant digits previously
1.88248    -3.25443E-7    determined.
SOLUTION IS X = 1.88248
-------------------------------------------------

SRU        0.066 UNTS.

RUN COMPLETE.
```

*Fig. 2. Search printout for $\sin(x+2)-e^x + x^2 + 3.701 = 0$.*

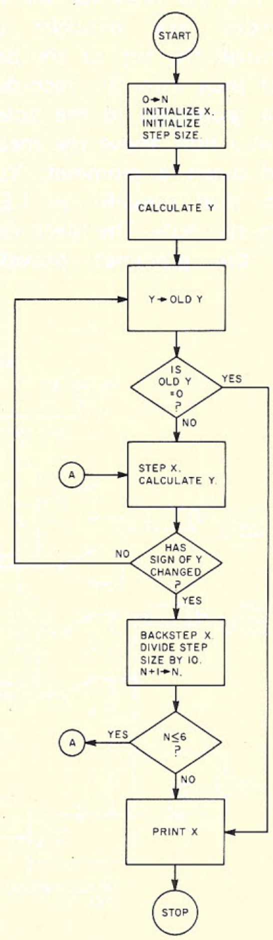



*Fig. 1. Program flowchart.*

0 is in between, and thus conclude that the solution is between x = 1 and x = 2. We can not only use the computer to search for the two values of x for which a sign change occurs, but we can also use the computer to further refine the solution.

If we stepped x through -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1 and finally detected the sign change at 2, we could then ask the computer to return to x = 1 and step by tenths until the sign changed again. By again reversing one step size and refining the step size to hundredths yet greater precision can be obtained. This refinement may be continued to an accuracy bounded only by the number of significant digits allowed by the language and compiler in use.

The sign change is easy to detect. If the previous value of y is multiplied by the present value, the sign will be positive as long as no sign change has occurred. When a sign change occurs, the product will be negative. The sign of this product may be used as a test for a sign change. Of course, one must also consider the possibility that at some stage of the computation y might actually be zero, so the product also must be tested for zero. Program A will implement the above method. (Fig. 1 is a flowchart of the program.)

If you want to solve the equation $\frac{x}{3} = 2x + 9$, merely change line 04000 to 04000 LET Y = X/3 - 2*X - 9. Similarly, other equations may be solved.

This method has certain weaknesses. If two solutions are close together, y may change signs twice in one step, causing you to miss both solutions. Also, if y goes from a positive number to zero and then immediately becomes positive again, that solution will be completely overlooked unless the computer happens to land exactly on it. Graphing the printout may help you realize when you have overlooked these types of roots. It is also difficult to find very large positive solutions and very small negative solutions due to the number of iterations necessary to locate the unit interval.

Fig. 2 is the printout of the solution to the equation $\sin(x + 2) - e^x + x^2 + 3.701 = 0$. The search was started at x = -11 with a step size of 1.

The equation x - 1 = 0 has an integer solution. It hardly needs a computer to solve it (Fig. 3) but it is shown here as an example when y becomes exactly 0. The solution

```
04000 LET Y = X - 1

RUN

 77/05/18. 09.08.35.
PROGRAM  KBAUD

INITIAL VALUE OF X AND STEP SIZE ARE? -11,1

X         Y

-11       -12
-10       -11            Search for unit interval.
-9        -10
-8        -9                13 iterations.
-7        -8
-6        -7             Computer finds exact answer
-5        -6
-4        -5             during this search.
-3        -4
-2        -3
-1        -2
0         -1
1         0
SOLUTION IS X = 1

SRU          0.040 UNTS.

RUN COMPLETE.
```

*Fig. 3. Solution for x - 1 = 0.*

```
04000 LET Y = X - .999999

RUN

 77/05/18. 09.10.13.
PROGRAM    KBAUD

INITIAL VALUE OF X AND STEP SIZE ARE? -11,1

X          Y

-11        -12.
-10        -11.
-9         -10.
-8         -9.                 Search for unit interval.
-7         -8.
-6         -7.                    13 iterations
-5         -6.
-4         -5.
-3         -4.
-2         -3.
-1         -2.
0          -.999999
1           .000001
.1         -.899999
.2         -.799999
.3         -.699999
.4         -.599999
.5         -.499999               10 iterations.
.6         -.399999
.7         -.299999
.8         -.199999
.9         -.099999
1.          .000001
.91        -.089999
.92        -.079999
.93        -.069999
.94        -.059999
.95        -.049999               10 iterations.
.96        -.039999
.97        -.029999
.98        -.019999
.99        -.009999
1.          .000001
.991       -.008999
.992       -.007999
.993       -.006999
.994       -.005999
.995       -.004999               10 iterations.
.996       -.003999
.997       -.002999
.998       -.001999
.999       -.000999
1.          .000001
.9991      -.000899
.9992      -.000799
.9993      -.000699
.9994      -.000599
.9995      -.000499               10 iterations.
.9996      -.000399
.9997      -.000299
.9998      -.000199
.9999      -.000099
1.          .000001
.99991     -.000089
.99992     -.000079
.99993     -.000069
.99994     -.000059
.99995     -.000049               10 iterations.
.99996     -.000039
.99997     -.000029
.99998     -.000019
.99999     -.000009
1.          .000001
.999991    -.000008
.999992    -.000007
.999993    -.000006
.999994    -.000005
.999995    -.000004               10 iterations.
.999996    -.000003
.999997    -.000002
.999998    -.000001
.999999    -9.23706E-14
1.          .000001
SOLUTION IS X = .999999

SRU         0.074 UNTS.

RUN COMPLETE.
```

*Fig. 4. Solution for x - .999999 = 0.*

is found during the search for the unit interval. This might be called a best case.

The equation x - .999999 = 0 does not look much different than the preceding example, but notice how much harder the computer works to solve this one (Fig. 4). This would definitely be a worst case.

In most solutions, we would expect the six digits to appear in a random fashion. This means that each digit will require, on the average, about five iterations. We can expect the number of iterations required to solve an equation to vary from 1 to 60 after the unit interval is found. The number of iterations required to find the unit interval depends upon how far from the solution you guess the initial value of x.

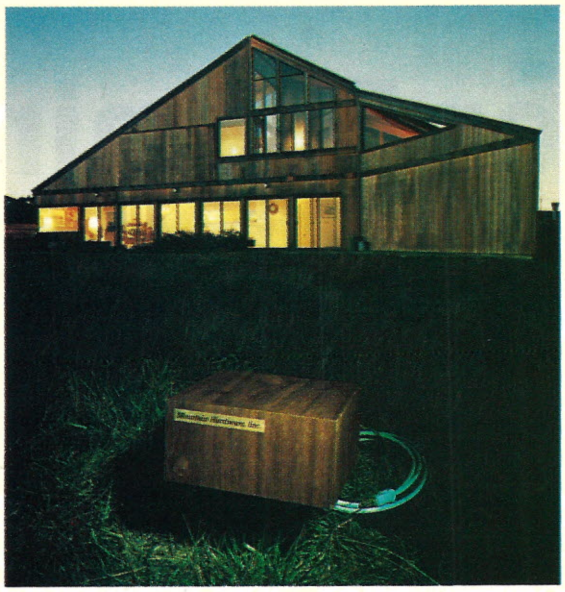
If you intend to solve x - 999999 = 0 on your computer, you'd better not start at x = -11, unless you just want to test the speed of your machine.

Happy crunching. ■

```
00100 REM  PROGRAM TO SOLVE AN EQUATION BY TRIAL AND ERROR.
00200 REM
00300 REM      MAP OF IDENTIFIERS.
00400 REM          N=COUNTER. COUNTS 1 EACH TIME ONE MORE DECIMAL IS
00500 REM              ADDED TO THE ACCURACY.
00600 REM          X=VALUE OF UNKNOWN IN THE EQUATION.
00700 REM          D=STEP SIZE, OR THE AMOUNT X IS TO BE INCREASED
00800 REM              AFTER A TRIAL HAS BEEN MADE.
00900 REM          Y=THE VALUE OF THE RIGHT MEMBER OF THE EQUATION.
01000 REM              THE EQUATION IS SOLVED WHEN Y REACHES ZERO OR
01100 REM              6 DECIMAL PLACES FOR X HAVE BEEN DETERMINED IN
01200 REM              ATTEMPT TO MAKE Y ZERO.
01300 REM          Y1=THE OLD VALUE OF Y.
01400 REM
01500 REM
01600 LET N = 0
01700 PRINT "INITIAL VALUE OF X AND STEP SIZE ARE";
01800 INPUT X,D
01900 PRINT
02000 PRINT
02100 PRINT "X","Y"
02200 PRINT
02300 GOSUB 04000
02400 LET Y1 = Y
02500 IF Y1 = 0 THEN 03300
02600 LET X = X + D
02700 GOSUB 04000
02800 IF Y1*Y >= 0 THEN 02400
02900 LET X = X - D
03000 LET D = D/10
03100 LET N = N + 1
03200 IF N <= 6 THEN 02600
03300 PRINT "SOLUTION IS X = ";X
03400 GOTO 04300
03500 REM
03600 REM              SUBROUTINE
03700 REM THE FIRST LINE OF THE SUBROUTINE MAY BE CHANGED IF DESIRED
03800 REM TO SOLVE A DIFFERENT EQUATION.
03900 REM
04000 LET Y = SIN(X + 2) - EXP(X) + X↑2 + 3.701
04100 PRINT X,Y
04200 RETURN
04300 END
```

*Program A. Trial and error method for solving an equation.*

# The Ultimate Turn-on



## On/off control everywhere-
## by computer over the AC wiring

Now it's simple and economical to control AC devices remotely from an S-100 or Apple II computer. Mountain Hardware's new Introl™ system delivers on/off commands over the existing AC lines — so you don't have to string a foot of wire!

**Control at any AC outlet.** The Introl system impresses a code-modulated 50 KHz control signal on the house wiring. Then decodes the signal at any outlet to switch AC devices on and off. You can control lights, refrigerators, TVs, solenoid valves, sprinklers, burglar alarms—and many other things we leave to your fertile imagination. With the addition of input sensors to your computer system, you can automatically control variables such as temperature and soil moisture.

**Here's how it works.** You plug in a single AC Controller board at the computer bus and connect the AC Interface Adapter to any convenient 115 VAC outlet. The AC Controller is now connected to address as many as 64 channels remotely. But it's completely isolated from the 115v power, so there's no chance of short or shock.

At any outlet where you seek control, plug in a Dual Channel AC Remote. Then plug one or two devices to be controlled into the box. Every AC remote has two independent 500 watt channels. When commanded by the computer, the Dual Channel AC Remote turns the devices on and off independently. When polled by the computer, the Dual Channel AC Remote sends a signal back, telling the computer the status of each device. Bidirectional communication provides error free operation.

**Simple programming.** You write your control program in BASIC or Assembler language. Software subroutines for the control programs come with the equipment — along with complete documentation. If you have an S-100 computer, you can program on/off commands at any day and time using our optional 100,000 day Calendar/Clock Board. A self contained power source assures fail safe operation.

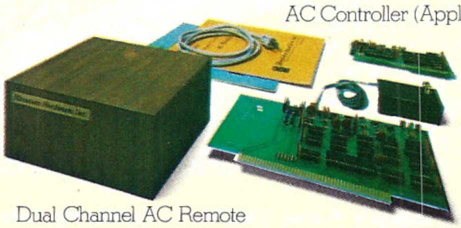
**Modest prices.** The AC Controller, for both the S-100 and Apple II computers, costs $149 in kit form or $189 completely assembled and tested. Each Dual Channel AC Remote costs $99 as a kit or $149 assembled and tested. Thus, a fully operative system in kit form can be yours for as little as $248.

The Calendar/Clock Board for S-100's costs $179 in kit form, $219 assembled and tested.

All prices are f.o.b. Ben Lomond, CA. Prices are USA Domestic. California residents add 6% sales tax.

**Where to find it.** The Introl system can now be found at computer shops throughout the U.S. and Canada. Drop by and ask for a demonstration. Mountain Hardware, Inc., may be reached at Box 1133, Ben Lomond, CA 95005. Phone (408) 336-2495.

AC Controller (Apple)



Dual Channel AC Remote

AC Controller (S-100)

**Mountain Hardware** M37

*R. R. Derynck*
*#28 336 Rundlehill Dr NE*
*Calgary Alberta*
*Canada T1Y 2Y2*

# Super Terminal!

## *interfacing the Burroughs 9350-2*

*This is a big article. Naturally, when it comes to publishing something of this size an evaluation is necessary to determine whether the material will be of value to a large number of people. There are four overriding considerations that have brought this article to the pages of* Kilobaud. *First, it is a beautifully prepared and detailed construction article that will provide more-than-adequate information for the experienced hobbyist attempting the project. Second, there are a lot of Burroughs 9350-2s in the field. Most of those released through the surplus channels were sold through Herbach & Rademan in Philadelphia. They told me that they had sold over 1500 of the units, and there were other companies selling them also. The third is the practical and unique approach Ron took in designing the interface to make the Burroughs look like a teletypewriter to his computer. Also, his interface replaces a whole cabinet of electronics and fits very neatly in the rear of the keyboard-printer. Last, the unit has a good reputation for reliability and is probably one of the smartest-looking terminals around. — John.*

*The author's Friden TM20K714 keyboard-printer after modification. This unit along with the TM20K715 controller make up the Burroughs 9350-2 communication terminal. (All photographs by Bob Padget.)*

About a year ago, I purchased one of the Burroughs 9350-2 communications terminals that became available on the surplus market at that time. This unit comes in two pieces: the Friden TM20K714 keyboard-printer and the TM20K715 controller. Unfortunately, absolutely no documentation is supplied with the terminal. However, being in need of a hard-copy I/O device for my computer system, I took a chance and ordered the unit despite this deficiency.

### Initial Checkout

When the terminal arrived I was quite impressed with the keyboard-printer's clean appearance and modern styling. I was somewhat disappointed that, although advertised as having both uppercase and lowercase characters, the lowercase characters were just small capitals rather than true lowercase as used on most typewriters.

The controller is a 9 x 17½ x 22½ inch box, which contained a power supply, 13 cards populated with obsolete DTL ICs (not 930 series) and what I presumed to be a delay line memory. The controller had obviously been serviced because a few of the cards had strips of masking tape applied to them, containing phrases such as "intermittent in decode logic bit 5!"

In spite of this discouraging indication I connected the keyboard-printer to the controller using the cables supplied (one for ac power to the controller, and the thicker multiconductor cable for dc power and control signals from the controller). It did not take long to discover that by connecting pin 5 to pin 20 on the 25-pin D connector and by

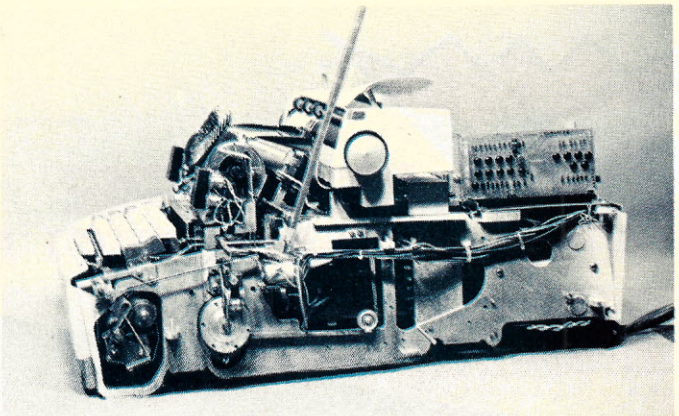
placing the line switch in the on position I could illuminate the ON LINE indicator. With a little more experimentation, I found that a message could be typed into the controller's memory and then printed on the terminal by the following procedure.

With pins 5 and 20 connected as described previously, the power switch on and the line switch off, temporarily depress the transmit switch; this should cause the transmit indicator to light. Now turn the line switch on and start typing. When the ETX or ETB switch is depressed, the characters just typed in will be printed out again. This test indicates that the keyboard-printer is working and that the controller's memory and terminal control circuits are functioning. It does not, however, test the RS 232B transmit and receive circuits or the controller's decoding logic. To do this the terminal must be connected to a computer's serial I/O port set up for the RS 232B communication standard (mark greater than +3 volts, space less than -3 volts). The procedure for doing this, and the software required to allow the terminal to communicate with an Altair/Imsai microcomputer, has been described in a previous article.[1]

### Operating Experience

I was able to get the Burroughs terminal to operate with my computer (a surplus BIT 483 minicomputer) in much the same manner as described in the above referenced article. However, I was not really satisfied with the way the system performed. For example, to type a line of data into the computer, it was necessary first to press the transmit key, then type the line and then press the ETX or ETB switch. For someone used to a Teletype or similar terminal, these extra steps are annoying. Pressing the transmit key causes an STX[2] character to be transmitted by the terminal. The ETX or ETB key sends the message plus an LRC character to the computer.

When transmitting data to the Burroughs terminal, it is necessary to frame each message, no matter how short,



*View of the keyboard-printer with right-hand dress panel removed. Pointer indicates the keyboard unlock solenoid, which may be mechanically defeated. The card with all the transistors on it, at the back of the terminal, contains the driver circuits shown schematically in Fig. 1.*

with the ASCII STX and ETX characters. In addition, an LRC error-detection character must be computed and transmitted immediately following the ETX character. A longitudinal redundancy check (LRC) is generated by Exclusive ORing all characters in the message to form a new or redundant character used to detect errors that may occur during transmission of a message.

When the terminal starts receiving, it also starts computing its own LRC. If this LRC is the same as that received from the computer, the terminal knows it got the message without any errors and sends an ASCII ACK character back to the computer and starts printing the message. If the two LRC characters are not equal, the terminal sends back a NAK character to indicate to the computer that it should retransmit the message.

Any software receiving or sending data to the Burroughs terminal must be able to handle the generation of all the special characters and must know what to do when it receives the special characters from the terminal. At best, this complicates the input and output routines, but at worst, in systems using PROM monitors for TTY I/O routines (such as SC/MP's KITBUG or Motorola's 6800 MIKBUG), the terminal will not operate at all.

For these reasons, I decided that I would modify my Burroughs terminal so it could be treated as a Teletype (TTY) by the software. Besides, in these days of LSI it just didn't seem right that the huge box of electronics (the controller) was required to make the terminal function! It seemed to me that all the required circuits could be built into the back of the keyboard-printer and the controller done away with completely. The remainder of this article describes how this objective has been met and provides designs for both parallel and serial (RS 232C) interfaces for the keyboard-printer.

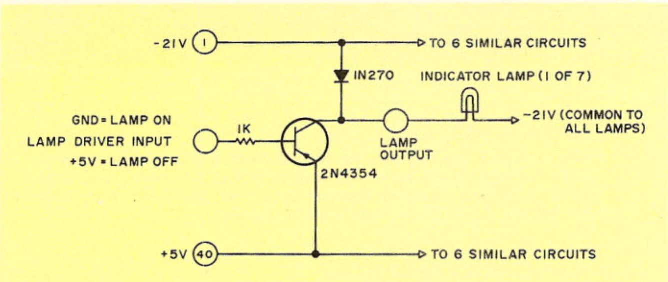



*Fig. 1a. Friden TM20K714 lamp driver circuit. (See Table 1 for input and output pin numbers.)*

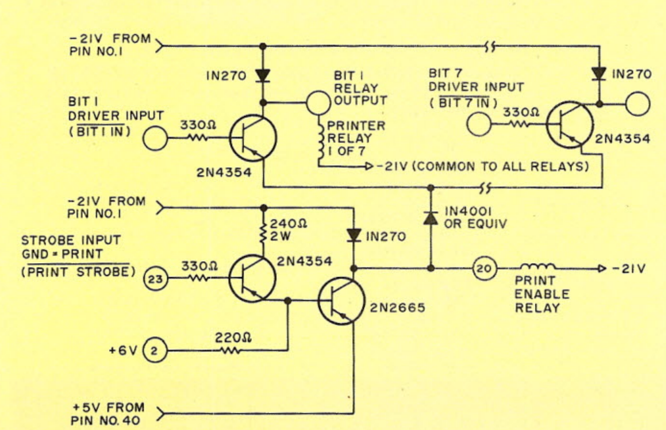



*Fig. 1b. Printer driver circuit. (Also see Table 1.)*

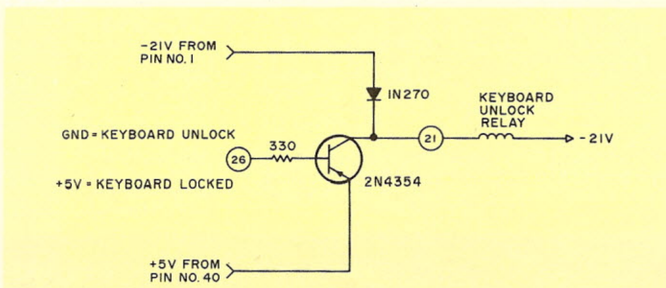
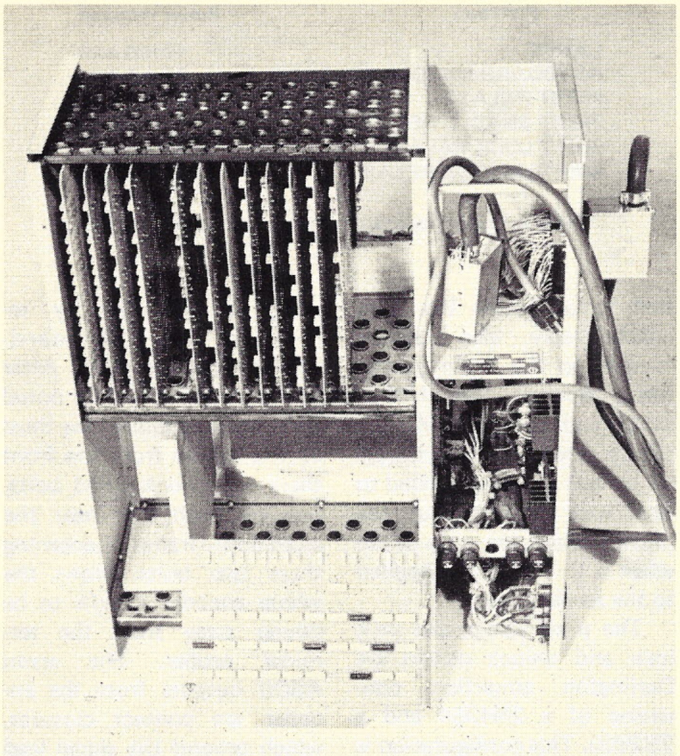
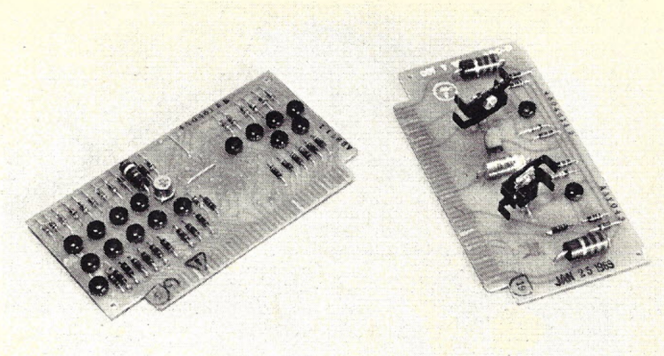



*Fig. 1c. Keyboard unlock driver.*



The TM10K715 controller with cover removed. One of the 13 logic boards is shown in front of the controller. It also contains a delay line memory and a power supply. The circuits described in this article completely replace this controller and are installed inside the keyboard-printer chassis.



*These two driver cards are part of the keyboard-printer as purchased. The card on the right is the shift lock and unlock driver (refer to Fig. 2). Note that the original 2N2665 transistors became a victim of my experiments and were replaced by TIP30 PNPs. The card on the left contains all the other driver circuits.*

| Card edge pin number | Main connector block pin number | Function |
|---|---|---|
| 1 | 76 | -21 V power supply |
| 2 | 79 | +6 V power supply |
| 3 | 50 | RECEIVE lamp driver input |
| 4 | 53 | SEND lamp driver input |
| 5 | -- | RECEIVE lamp output |
| 6 | 52 | TRANSMIT lamp driver input |
| 7 | -- | SEND lamp output |
| 8 | 51 | BUFFER OV/FLO lamp driver input |
| 9 | -- | TRANSMIT lamp output |
| 10 | 62 | TIME OUT lamp driver input |
| 11 | -- | BUFFER OV/FLO lamp output |
| 12 | 54 | ERROR lamp driver input |
| 13 | -- | TIME OUT lamp output |
| 14 | -- | ERROR lamp output |
| 20 | -- | PRINT ENABLE output |
| 21 | -- | KEYBOARD UNLOCK output |
| 22 | -- | BIT 2 relay output |
| 23 | 48 | PRINT ENABLE driver input |
| 24 | -- | BIT 3 relay output |
| 25 | -- | BIT 1 relay output |
| 26 | 56 | KEYBOARD UNLOCK driver input |
| 27 | -- | BIT 4 relay output |
| 28 | 42 | BIT 2 driver input |
| 29 | -- | BIT 5 relay output |
| 30 | -- | BIT 6 relay output |
| 31 | 43 | BIT 3 driver input |
| 32 | -- | BIT 7 relay output |
| 33 | 41 | BIT 1 driver input |
| 34 | 55 | ON LINE driver input |
| 35 | 44 | BIT 4 driver input |
| 36 | 45 | BIT 5 driver input |
| 37 | 46 | BIT 6 driver input |
| 38 | -- | ON LINE lamp output |
| 39 | 47 | BIT 7 driver input |
| 40 | 78 | +5 V power supply |

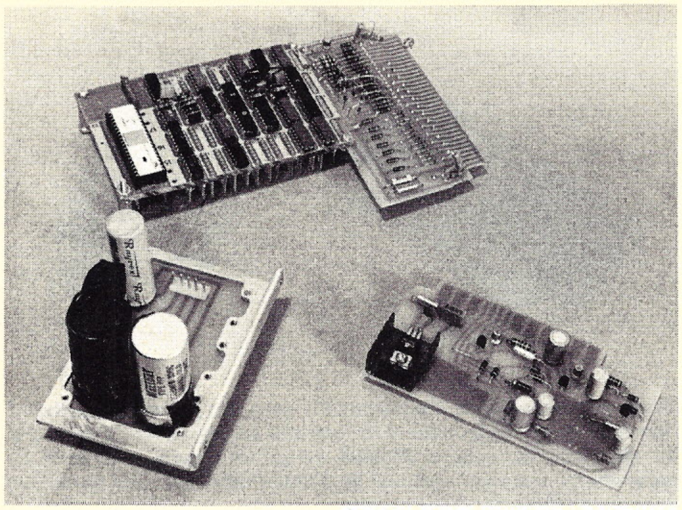
*Table 1. Friden keyboard-printer lamp and relay drive card pinouts.*

## Operation of the TM20K714 Keyboard-Printer

When I started to work on the printer, I was pleasantly surprised to discover that it is designed to accept, decode and print ASCII characters. Logically, then, the keyboard should generate the 7-bit ASCII code as well. It took a while to figure out how this is accomplished, but it turns out that the keyboard-printer generates and prints the 7-bit ASCII code in parallel fashion. Therefore, once the various signal wires are identified it will be relatively easy to modify the terminal for either serial or parallel ASCII communications.

Before designing new control electronics and modifying the terminal, you must study and understand the functions of the input and output circuits located inside the case of the keyboard-printer. These circuits are set up to accept or generate the 7-bit ASCII code set. The input circuits are transistor drivers located on two printed circuit cards at the back of the terminal. The output circuits are contact closures indicating that a key has been pressed, or the current status of the terminal (e.g., carriage in uppercase position). Refer to the photographs for pictures of these circuits. The next section of the article describes these input and output circuits. Following sections describe a power supply for the terminal and new control electronics that can be added inside the case of the keyboard printer. Adding the new control section to the existing input/output circuits will provide a terminal with the following characteristics.

- RS 232C compatible.
- Prints both uppercase and



*The two cards in the foreground are the author's power-supply boards (Fig. 4). The wire-wrap board at the back contains the control circuits described in this article. These three circuit boards replace the TM20K715 controller.*

| Function | Power required |
|---|---|
| LAMPS | 40 mA each |
| KEYBOARD UNLOCK SOLENOID | 95 mA each |
| PRINT RELAYS | 180 mA each |
| PRINT ENABLE RELAY | 300 mA each |
| SHIFT LOCK RELAY | 1500 mA each |
| SHIFT UNLOCK RELAY | 1500 mA each |

*Table 2. Friden keyboard-printer power requirements.*

lowercase characters.

- Keyboard generates all uppercase, lowercase and control character codes.
- Serial communications at 110 baud.

Once the new control electronics are built the TM20K715 controller is no longer needed.

**Input**

All input signals to the terminal are ground true TTL logic levels (i.e., ground or 0 V equals function performed, +5 V equals function not performed). These signals are input to transistor driver circuits located on the two PC cards at the back of the terminal. The card with the greatest number of transistors on it contains the drivers for all the indicator lights (except power on), the keyboard unlock solenoid, the print strobe and the seven ASCII inputs. The power-on light is illuminated when the on/off switch is in the on position. These driver circuits are shown in Fig. 1, and Table 1 lists the pinouts of this card. The second card, with heat sinks on two of the transistors, contains the drivers for the shift lock and unlock relays. A schematic of this card is shown in Fig. 2. Normally, the controller supplies the required -21 V, +5 V and +6 V power supplies. All lamps and relays are connected between the -21 V and +5 V power supplies when the driver transistors are turned on.

The circuits shown in Fig. 1 are all simple transistor-switching circuits, with the emitter of the PNP transistors connected through the load (relay or lamp) to -21 V. To turn on the transistor, and thus supply current to the load, it is necessary to switch the input from +5 V to ground (0 V). The 1N270 diodes in parallel with the load are used to protect the transistors from voltage spikes that occur when current is switched off in an inductive load such as a relay coil.

The strobe input, in addition to activating the print enable relay, also supplies current to the seven printer relays through the 1N4001 diode and the 2N4354 transistors when they are turned on by a 0 V signal applied to their base circuits. Thus, the printer relays are only active when a 0 V signal is applied to the strobe input.

The print strobe and shift lock and unlock drivers are Darlington amplifiers consisting of a 2N4354 and a 2N2665. This configuration is required on these circuits because of the high currents they are required to switch. Table 2 lists the current requirements for lamps and relays used in the keyboard-printer. The shift lock and unlock functions are latching, and it is only necessary to pulse one of these inputs for 10 to 20 milliseconds to change the printer from shift lock to unlock or vice versa. The print strobe input also requires a pulse of 10 to 20 milliseconds' duration.

**Output**

The terminal's keyboard is encoded to ASCII characters by a matrix of levers and switches located at the bottom of the terminal. Access to this mechanical encoder is obtained by removing the keyboard-printer's bottom cover. After this is done, it will be noted that approximately one-third of the way in from the front there are two knurled bolts, one on each side near the aluminum frame. Loosening these two bolts allows the whole encoder matrix to be swung away from the terminal proper. The seven ASCII outputs from the encoder are contact closures, which ground the signal lead when active (see Fig. 3). These seven leads by themselves do not provide the proper outputs for control characters, however. Bits 1 and 4 are not encoded properly when the Control Case button is depressed. There are a number of contacts in the terminal associated with the Control Case function, and two of these can be ORed in with the normal Bit 1 and 4 outputs to properly encode the control characters. Therefore the keyboard is capable of generating the 7-bit ASCII code for uppercase and lowercase characters as well as all the control characters indicated in red letters on the edge of the terminal's keys. Additional SPDT contacts are provided (refer to Fig. 3) to

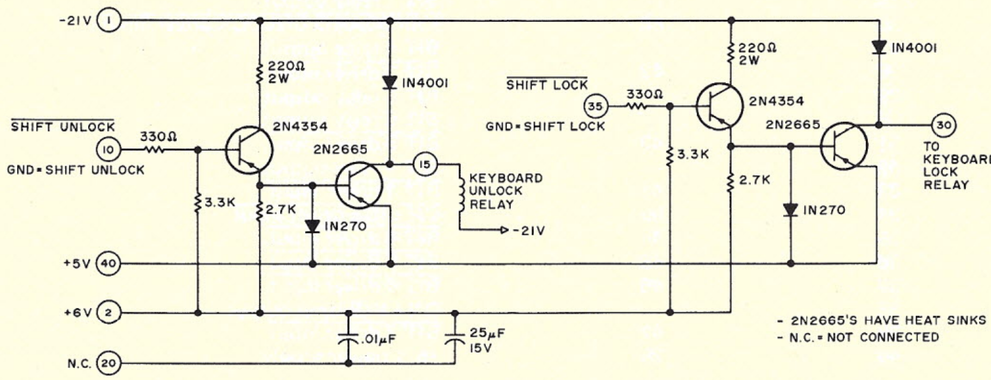



*Fig. 2. Shift lock and unlock driver circuit.*

| Status switch name | Main connector block pin number N.C. | N.O. | Function |
|---|---|---|---|
| CRET | 4 | 3 | Operates during carriage return. |
| SLCK | 7 | 5 | Operates when keyboard shift is locked. |
| ETB | 18 | 17 | Operates when ETB switch depressed. |
| ETX | 16 | 15 | Operates when ETX switch depressed. |
| ONLN | 11 | 12 | Operates when LINE switch on. |
| SHFT | 2 | 1 | Operates when SHIFT key depressed. |
| XMIT | 14 | 13 | Operates when TRANSMIT switch depressed. |
| RSET | 60 | 38 | Operates when RESET switch depressed. |
| PRNT | 10 | 8 | Operates when character is printed. |
| KYPR | 21 | 20 | Operates when any key pressed. |

Note: when switch operates, the N.O. contact goes to ground and the N.C. contact switches off ground.

*Table 3. Friden keyboard-printer status-switch functions.*

indicate functions such as any key being pressed, carriage in motion and various other functions, as outlined in Table 3.

All input, output and power wires in the keyboard-printer are connected to the main terminal block located on the bottom right-hand side of the terminal at the back of the unit. I have figured out the functions of all but a few of the unimportant pins on this main connector block, as listed in Table 4. Fig. 3 illustrates the Friden keyboard-printer in block diagram format. It should be noted that although the printer accepts the 7-bit ASCII code, the printer does not shift to print both uppercase and lowercase characters as defined in this code. For example, if the printer is in the shift-lock position, a capital A will be printed when either the ASCII code for A (11 000 001) or the ASCII code for a (11 100 001) is received. Similarly, if the printer is in the shift-unlock position, a will be printed when either 11 000 001 or 11 100 001 is received. Therefore, if the terminal is to be used to print the full upper and lower ASCII character set, the shift lock and unlock signals must be generated by the control electronics added to the terminal.

Another problem with the terminal is that the symbols , - . and /, which are defined to be uppercase (i.e., shift locked) in the ASCII code, are grouped with the lowercase characters. Similarly, the characters < = > and ?, which should be lowercase, are grouped with the uppercase characters. These problems do not occur when these keys are encoded by the terminal. Therefore, they need only to be considered when decoding the incoming ASCII data to generate the shift lock and unlock pulses.

### Design Considerations for the New Control Electronics

Before starting the design of new control electronics for the keyboard-printer you should thoroughly study the block diagram of Fig. 3 and the data in Table 4. You will note that the terminal is quite flexible. For example, by studying Fig. 3, you'll note that two output ports from a microcomputer could be used to: I. turn on all of the terminals indicator lamps; 2. lock or unlock the keyboard; 3. perform the shift lock and unlock functions as well as supply the 7-bit ASCII data to be printed.

Similarly, two input ports could be used to: 1. read in the seven ASCII bits generated when a key is pressed and 2. read in the nine switches indicating terminal status. The 10th status switch KYPR (key pressed) is used to interrupt the microcomputer when a key is activated.

For serial interfaces, a UART and an RS 232C or current loop driver circuit could be added to allow the Friden keyboard-printer to act as a Teletype or other serial ASCII terminal. The circuits described in the next section will allow the terminal to be hooked up to a computer in either a serial or parallel fashion.

Probably the simplest way to build a new controller would be to mount it in a separate box and connect it to the keyboard-printer's main connector block by the cable originally supplied with the unit. Another possibility, and the one I prefer, is to build the controller into the back of the keyboard-printer beside the two existing driver cards. There is room for six to ten 2 3/4 x 5 1/4 inch printed circuit boards in this area. Room for a new power transformer can be obtained by removing the main connector block and the connector used to supply 117 V ac to the old controller. This design would provide a fully self-contained terminal.

### Steps to Rebuild the Terminal

Assuming that the control

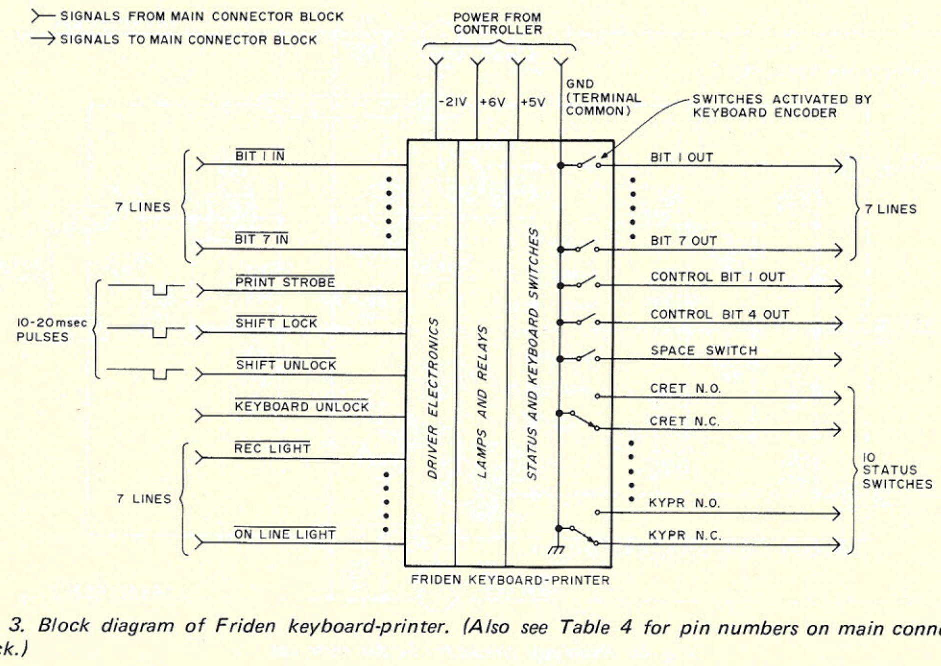



*Fig. 3. Block diagram of Friden keyboard-printer. (Also see Table 4 for pin numbers on main connector block.)*

electronics are to be built into the back of the Friden keyboard-printer, the following steps will be required in order to make the terminal operational.

1. Remove the main connector block and 117 V ac power connector from the keyboard-printer. The pins can be removed from the plastic housing of the main connector block using the AMP terminal-removal tool #305183. As each lead is removed from the connector block, it should be labeled with its pin number and/or function according to Table 4. (Note that Fig. 9 shows the original power wiring schematic.)

2. Build and install the power-supply circuit.

3. Build and install the parallel interface circuits.

4. If required, build and install the serial RS 232C interface.

**Terminal Power Supply Design**

As shown in Fig. 3 the keyboard-printer uses three power supplies: -21 V, +5 V and +6 V. The power supply is used only on the emitters of the Darlington transistor amplifiers used for print strobe, shift lock and shift unlock (see Figs. 1 and 2). These circuits will operate satisfactorily at +5 V instead of +6 V, thus simplifying the power-supply design by eliminating one of the required voltage levels.

In the original circuit design, the lamps and relays require a 26 V supply, which is obtained by wiring them between the -21 V and +5 V supplies. If a UART is employed for the serial interface it will require -12 V. The RS 232C or current loop interfaces require both +12 V and -12 V. It was found that the lamps and relays will work when supplied with 24 V. Therefore, rather than design a power supply for four voltages (-21 V, -12 V, +5 V and +12 V), it was decided to operate the keyboard-printer from a +12 V and -12 V power supply rather than the -21 V and +5 V used in the original design. Therefore, the power supply need only be designed to provide -12 V, +5 V and +12 V. The disadvantage of this approach is that the driver circuits of Figs. 1 and 2 will no longer be TTL compatible. However, they can be driven directly by 7406 or 7407 TTL open collector circuits that have output transistors rated for 30 V. Since the proposed terminal controller design does not allow printing and shifting to occur at the same time, the worst case current requirement for the keyboard-printer will be 1.6 A when either the shift lock or unlock relay is activated plus the current required by the lamps and control electronics (refer to Table 2). Therefore, a 25.2 VCT 2 A transformer such as Radio Shack #273-1512 can be used to power up both the terminal and the control electronics. Fig. 4 illustrates the suggested power-supply circuit.

This power supply may be broken into three sections:

1. A low current ±12 V supply for the serial interface card. This is a simple zener regulator consisting of $R_1$ and $Z_1$ for +12 V and $R_2$, $Z_2$ for -12 V.

2. A high current ±12 V supply[3] used to power the terminal's lamps, relays and driver circuits.

3. A +5 V supply using an LM340T-5 three-terminal voltage regulator to regulate the +12 V high-current supply down to +5 V for the TTL ICs.

**Parallel Interface — Input Section**

The input section of the parallel interface is shown in Fig. 5. Timing for the printer is derived from four TTL monostables consisting of one-half of a 74123 IC each. The first monostable generates a ten ms shift pulse (SHFTP), which is used to activate the shift lock or unlock drivers depending on the

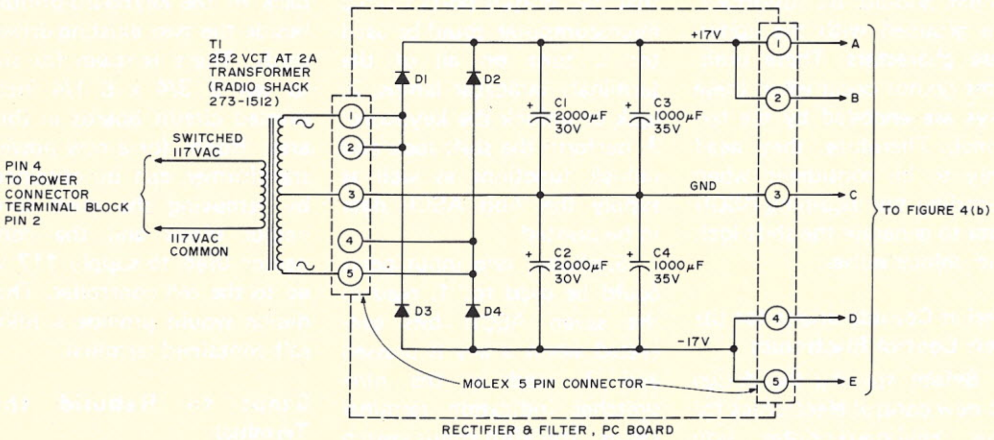



*Fig. 4a. Rectifier and filter for Friden terminal power supply.*

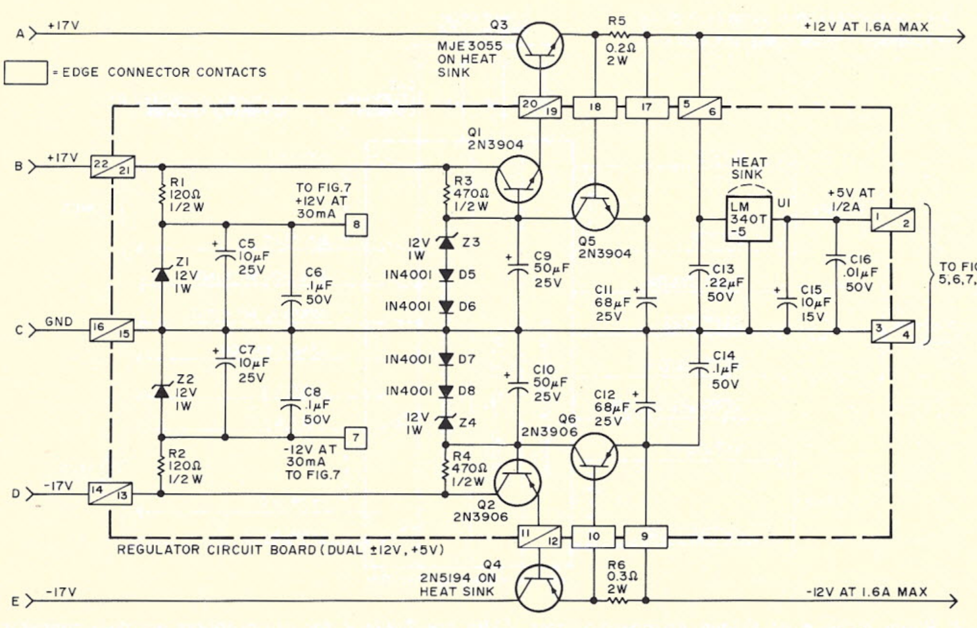



*Fig. 4b. Regulator circuit for Friden terminal.*

signals on the shift up (SHFTUP) and shift down (SHFTDN) inputs and on the current condition of the printer as determined by the shift-lock status switch, which is debounced by a pair of 7400 NAND gates.

The second monostable generates a six ms delay, which allows the printer time to settle in its new (locked or unlocked) position. At the end of this time period, the third monostable generates a ten-millisecond print strobe pulse, which causes the character represented by the $\overline{B1I}$ through $\overline{B7I}$ lines to be printed. The fourth monostable generates a 1 usec pulse to indicate the end of the print cycle (ENDPT).

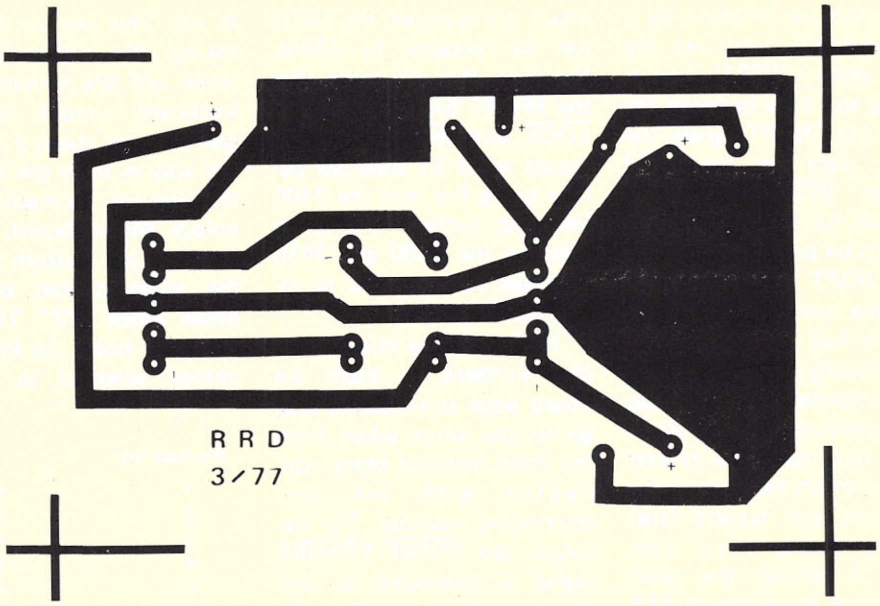



*Fig. 4c. Full-size printed circuit foil pattern for the rectifier-filter PC board.*

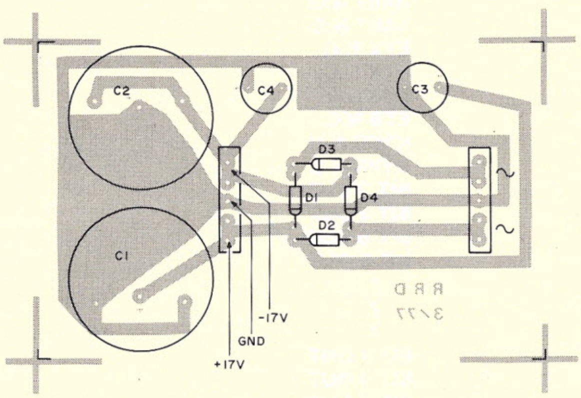



*Fig. 4d. Rectifier-filter PC board component placement.*

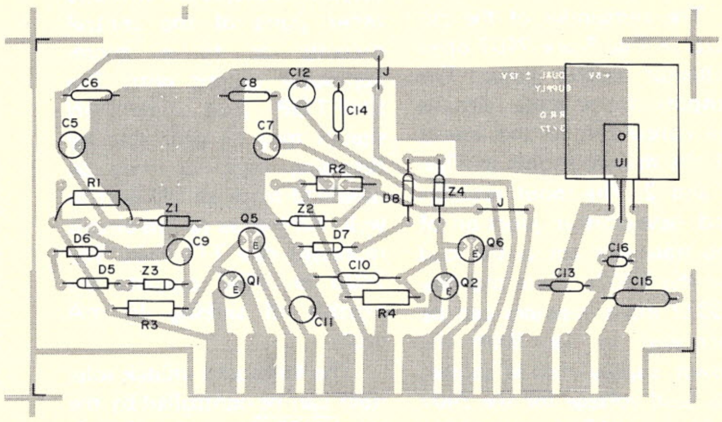



*Fig. 4f. Regulator circuit component placement diagram.*

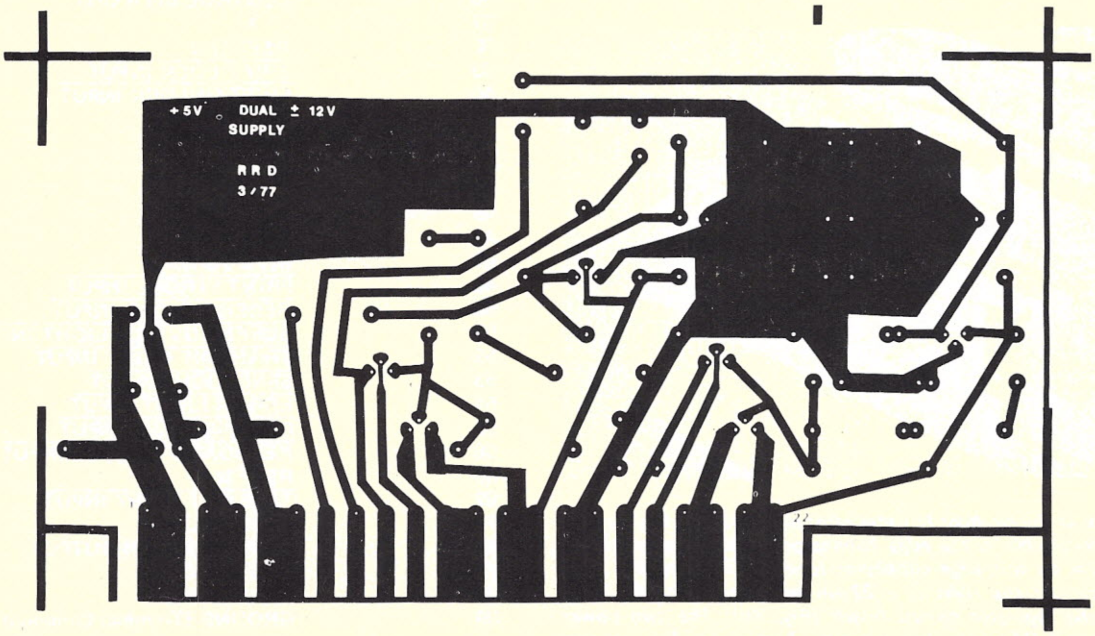



*Fig. 4e. Full-size printed circuit foil pattern for +5 V, +12 V, -12 V, regulator circuit.*

Printing is initiated by a positive-going pulse on the start print (STPT) input. During the time between the STPT and ENDPT signals, the ASCII data on lines $\overline{\text{B1}}$ through $\overline{\text{B7}}$ must remain constant for the proper character to be printed. Although the ENDPT signal indicates that the control electronics has finished the print cycle mechanically, the printer has not completed its cycle. By experimenting, I found that I could not make the printer operate much faster than ten characters per second (cps) without losing some characters. I believe the print mechanism is capable of 13.5 cps, even though the original controller communicated at 150 bits per second (15 cps).

The remainder of the circuits in Fig. 5 are 7407 open collector buffer gates. The outputs from these circuits are connected to the inputs of the drivers shown in Figs. 1 and 2. The input resistors and base-emitter junction of the transistor act as the load for the 7407s. Note that the 330 Ω input resistors can be increased to 1k. With the power supply that is shown, the load voltage for the 7407 buffers is +12 V, well within their 30 V rating. The inputs to the 7407s are ground-true TTL levels. If positive-true inputs are required, the 7407s can be replaced by 7406s. However, this will mean that the $\overline{\text{PRINT STROBE}}$, $\overline{\text{SHIFT LOCK}}$, and $\overline{\text{SHIFT UNLOCK}}$ signals must be inverted before being fed into the 7406 inverting buffers. The extra gates in the 7400 and 7410 packages can be used to invert these signals.

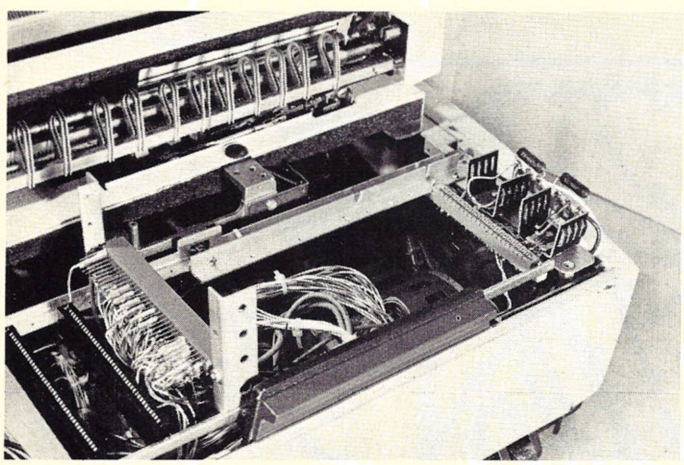
In Fig. 5, the circles with the numbers in them are board edge connections that go to the wires taken from the main terminal block connector with the corresponding number. For example, the $\overline{\text{PRINT STROBE}}$ signal is connected to wire that came from pin 48 on the main connector block. The card edge connections indicated by squares are wired to other parts of the control circuits or to a microcomputer output port. The SHFTUP and SHFTDN signals may be generated by decoding logic or may be obtained from an I/O port as well. All inputs to this circuit represent one TTL load. The 7406 or 7407 outputs are capable of sinking 40 mA each.

The keyboard unlock solenoid can be controlled by the the $\overline{\text{KYBDU}}$ signal in Fig. 5. However, it is more convenient and uses less power to mechanically unlock the keyboard. This can be done by taking the right-side dress panel off the terminal. The keyboard unlock solenoid will be seen about a third of the way in from the front of the terminal. To mechanically unlock the keyboard, a small tab near the bottom edge of the solenoid can be bent down about 1/8". This permanently holds the keyboard unlock solenoid in its unlocked position.

**Parallel Interface — Output Section**

The output section of the parallel interface is shown in Fig. 6. It consists of nine pairs of 7400 TTL NAND gates, which debounce the nine status switches in the terminal.

The key pressed signal (KYPR), which is generated



*This is the back of the keyboard-printer after modification. On the left, mounted horizontically, is the edge connector for the control circuit board. The wires on this edge connector formerly went to the main connector block. At the right is a 22-pin edge connector mounted vertically for the regulator circuit board (Fig. 4b). The two power transistors Q3 and Q4 are mounted on the heat sinks. The resistors behind the heat sinks are R5 and R6.*

| Pin number | Function |
|---|---|
| 1 | GND when SHIFT pressed (SHFT N.O.). |
| 2 | GND when SHIFT not pressed (SHFT N.C.). |
| 3 | CRET N.O. |
| 4 | CRET N.C. |
| 5 | GND when SHIFT LOCK. |
| 7 | GND when SHIFT UNLOCK. |
| 8 | GND when character being printed. |
| 10 | GND when character not being printed. |
| 11 | GND when LINE switch OFF. |
| 12 | GND when LINE switch ON. |
| 13 | XMIT N.O. |
| 14 | XMIT N.C. |
| 15 | ETX N.O. |
| 16 | ETX N.C. |
| 17 | ETB N.O. |
| 18 | ETB N.C. |
| 20 | KYPR N.O. |
| 21 | KYPR N.C. |
| 22 | BIT 2 OUT |
| 23 | BIT 3 OUT |
| 24 | BIT 6 OUT |
| 25 | BIT 5 OUT |
| 26 | ? |
| 27 | ? |
| 28 | ? |
| 29 | BIT 1 OUT |
| 30 | BIT 4 OUT |
| 31 | BIT 7 OUT |
| 32 | ? |
| 33 | ? |
| 34 | ? |
| 35 | CONTROL BIT 1 OUT |
| 36 | CONTROL BIT 4 OUT |
| 37 | ? |
| 38 | RSET N.O. |
| 39 | $\overline{\text{SHIFT LOCK INPUT}}$ |
| 40 | $\overline{\text{SHIFT UNLOCK INPUT}}$ |
| 41 | $\overline{\text{BIT 1 INPUT}}$ |
| 42 | $\overline{\text{BIT 2 INPUT}}$ |
| 43 | $\overline{\text{BIT 3 INPUT}}$ |
| 44 | $\overline{\text{BIT 4 INPUT}}$ |
| 45 | $\overline{\text{BIT 5 INPUT}}$ |
| 46 | $\overline{\text{BIT 6 INPUT}}$ |
| 47 | $\overline{\text{BIT 7 INPUT}}$ |
| 48 | $\overline{\text{PRINT STROBE INPUT}}$ |
| 50 | $\overline{\text{RECEIVE LIGHT INPUT}}$ |
| 51 | $\overline{\text{BUFFER OV/FLO LIGHT INPUT}}$ |
| 52 | $\overline{\text{TRANSMIT LIGHT INPUT}}$ |
| 53 | $\overline{\text{SEND LIGHT INPUT}}$ |
| 54 | $\overline{\text{ERROR LIGHT INPUT}}$ |
| 55 | $\overline{\text{ON LINE LIGHT INPUT}}$ |
| 56 | $\overline{\text{KEYBOARD UNLOCK INPUT}}$ |
| 60 | RSET N.C. |
| 62 | $\overline{\text{TIME OUT LIGHT INPUT}}$ |
| 76 | -21 V INPUT |
| 77 | -21 V POWER ON OUTPUT |
| 78 | +5 V INPUT |
| 79 | +6 V INPUT |
| 80 | GROUND (Terminal Common) |

*Table 4. Main connector block functions.*

every time any key on the keyboard is depressed, triggers a 74123 monostable that provides an 8 ms delay. When this delay expires a 1 usec pulse is generated by the second half of the 74123. This pulse (VLDKY) indicates that the data on the output lines B10 through B70 is valid. When a key is pressed, its corresponding ASCII code switch is closed for 15 to 20 ms. The 8 ms delay provided by the first half of the 74123 debounces the ASCII encoder matrix by ensuring that the matrix switch outputs are considered valid only for a brief interval that occurs halfway between the time the switch closes and the time it opens again.

Since the terminal's printer and keyboard are interlocked, VLDKY will be generated every time a key is activated either by pressing a key switch or by transmitting data from the computer to the terminal. The signal PRTCMP (print completed) is low only when a transmitted character is being printed. Therefore, it is ANDed with VLDKY to produce an output strobe, STBOUT, which is active only when a key is pressed, but not when the terminal is printing a character received from the computer. This stops the Burroughs terminal from echoing each character sent to it by the computer. Incidentally, because the keyboard and printer are mechanically interlocked, this terminal cannot be used as a full duplex device. In a full duplex terminal, the keyboard and printer are completely independent. When a key is pressed, the character is trans-

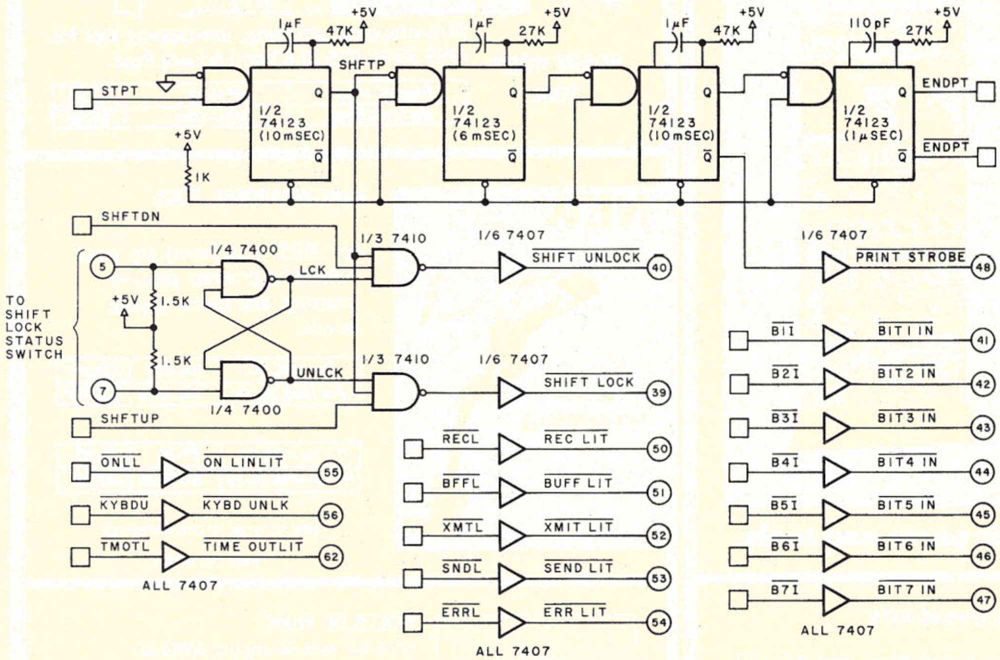



*Fig. 5. Parallel interface circuit, input section.*

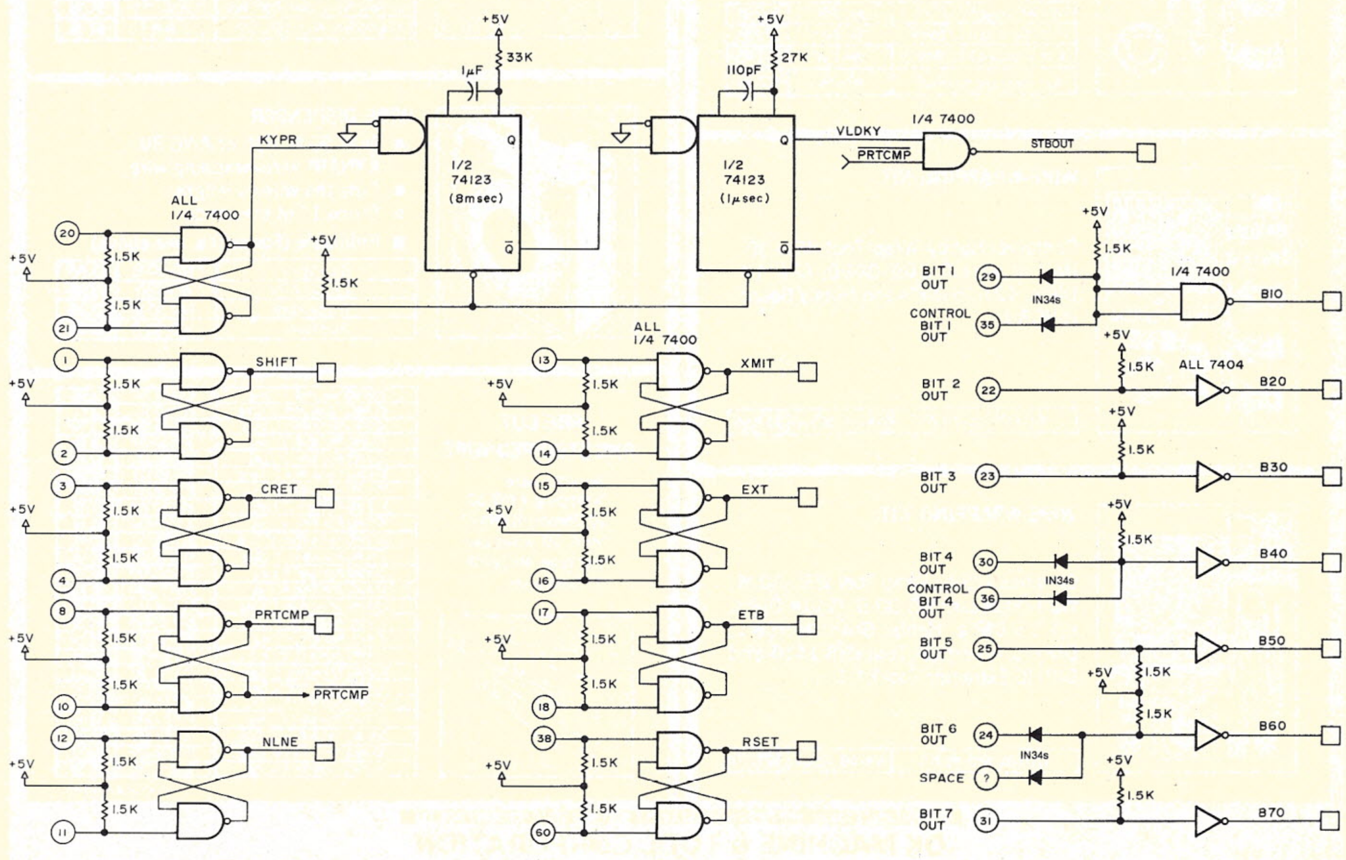
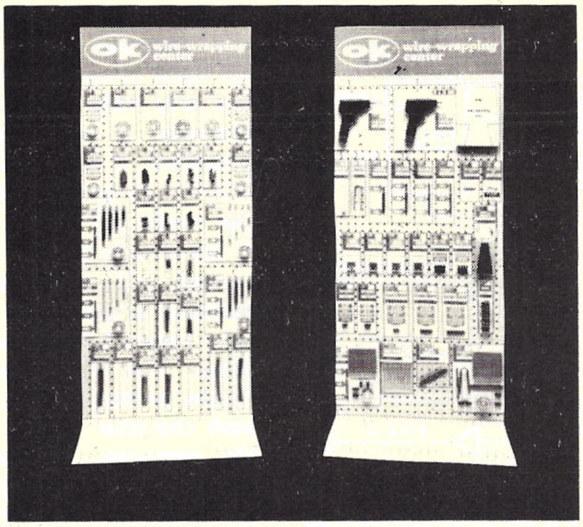
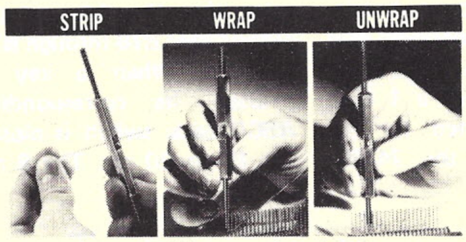


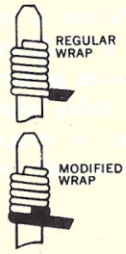

*Fig. 6. Parallel interface circuit, output section.*

# ok® wire wrapping center



**for quality electronic parts and tools.**




### HOBBY WRAP TOOL

Wire-wrapping, stripping, unwrapping tool for AWG 30 on.025 (0,63mm) Square Post.

| | | |
|---|---|---|
| Regular Wrap | WSU-30 | $6.95 |
| Modified Wrap | WSU-30M | $7.95 |

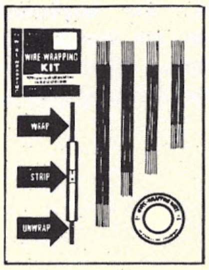

### WIRE-WRAPPING TOOL

For .025" (0,63mm) sq. post "MODIFIED" wrap, positive indexing, anti-overwrapping device.

| | | |
|---|---|---|
| For AWG 30 | BW-630 | $34.95* |
| For AWG 26-28 | BW-2628 | $39.95* |

| | | |
|---|---|---|
| Bit for AWG 30 | BT-30 | $3.95 |
| Bit for AWG 26-28 | BT-2628 | $7.95 |

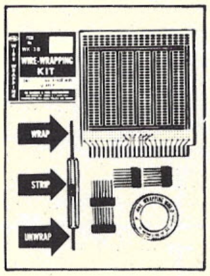
*USE "C" SIZE NI-CAD BATTERIES (NOT INCLUDED)



### WIRE-WRAPPING KITS

Contains: Hobby Wrap Tool WSU-30, (50 ft.) Roll of wire Prestripped wire 1" to 4" lengths (50 wires per package) stripped 1" both ends.

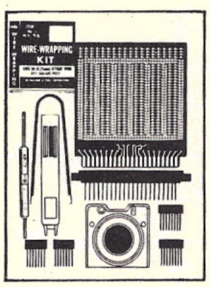
| | | |
|---|---|---|
| Wire Wrapping Kit. (Blue) | WK-2-B | $12.95 |
| Wire Wrapping Kit. (Yellow) | WK-2-Y | $12.95 |
| Wire Wrapping Kit. (White) | WK-2-W | $12.95 |
| Wire Wrapping Kit (Red) | WK-2-R | $12.95 |



### WIRE-WRAPPING KIT

Contains: Hobby Wrap Tool WSU-30, Roll of wire R-30B-0050, (2) 14 DIP's, (2) 16 DIP's and Hobby Board H-PCB-1.

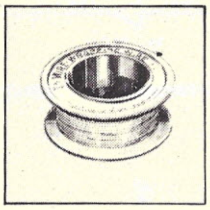
| | | |
|---|---|---|
| Wire-Wrapping Kit | WK-3B (Blue) | $16.95 |



### WIRE-WRAPPING KIT

Contains: Hobby Wrap Tool WSU-30 M, Wire Dispenser WD-30-B, (2) 14 DIP's, (2) 16 DIP's, Hobby Board H-PCB-1, DIP/IC Insertion Tool INS-1416 and DIP/IC Extractor Tool EX-1

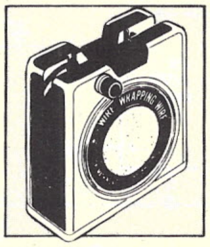
| | | |
|---|---|---|
| Wire-Wrapping Kit | WK-4B (Blue) | $25.99 |



### ROLLS OF WIRE

Wire for wire-wrapping AWG-30 (0.25mm) KYNAR® wire, 50 ft. roll, silver plated, solid conductor, easy stripping.

| | | |
|---|---|---|
| 30-AWG Blue Wire, 50ft. Roll | R-30B-0050 | $1.98 |
| 30-AWG Yellow Wire 50ft. Roll | R-30Y-0050 | $1.98 |
| 30-AWG White Wire, 50ft. Roll | R-30W-0050 | $1.98 |
| 30-AWG Red Wire, 50ft. Roll | R-30R-0050 | $1.98 |



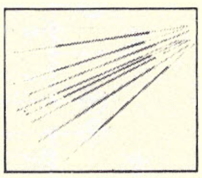
### WIRE DISPENSER

- With 50 ft. Roll of AWG 30 KYNAR® wire-wrapping wire.
- Cuts the wire to length.
- Strips 1" of insulation.
- Refillable (For refills, see above)

| | | |
|---|---|---|
| Blue Wire | WD-30-B | $3.95 |
| Yellow Wire | WD-30-Y | $3.95 |
| White Wire | WD-30-W | $3.95 |
| Red Wire | WD-30-R | $3.95 |

### PRE CUT PRE STRIPPED WIRE

Wire for wire-wrapping, AWG-30 (0.25mm) KYNAR® wire, 50 wires per package stripped 1" both ends.



| | | |
|---|---|---|
| 30-AWG blue Wire, 1" Long | 30-B-50-010 | $.99 |
| 30-AWG Yellow Wire, 1" Long | 30-Y-50-010 | $.99 |
| 30-AWG White Wire, 1" Long | 30-W-50-010 | $.99 |
| 30-AWG Red Wire, 1" Long | 30-R-50-010 | $.99 |
| 30-AWG Blue Wire, 2" Long | 30-B-50-020 | $1.07 |
| 30-AWG Yellow Wire, 2" Long | 30-Y-50-020 | $1.07 |
| 30-AWG White Wire, 2" Long | 30-W-50-020 | $1.07 |
| 30-AWG Red Wire, 2" Long | 30-R-50-020 | $1.07 |
| 30 AWG Blue Wire, 3" Long | 30-B-50-030 | $1.16 |
| 30-AWG Yellow Wire, 3" Long | 30-Y-50-030 | $1.16 |
| 30-AWG White Wire, 3" Long | 30-W-50-030 | $1.16 |
| 30-AWG Red Wire, 3" Long | 30-R-50-030 | $1.16 |
| 30-AWG Blue Wire, 4" Long | 30-B-50-040 | $1.23 |
| 30-AWG Yellow Wire, 4" Long | 30-Y-50-040 | $1.23 |
| 30-AWG White Wire, 4" Long | 30-W-50-040 | $1.23 |
| 30-AWG Red Wire, 4" Long | 30-R-50-040 | $1.23 |
| 30-AWG Blue Wire, 5" Long | 30-B-50-050 | $1.30 |
| 30-AWG Yellow Wire, 5" Long | 30-Y-50-050 | $1.30 |
| 30-AWG White Wire, 5" Long | 30-W-50-050 | $1.30 |
| 30-AWG Red Wire, 5" Long | 30-R-50-050 | $1.30 |
| 30-AWG Blue Wire, 6" Long | 30-B-50-060 | $1.38 |
| 30-AWG Yellow Wire, 6" Long | 30-Y-50-060 | $1.38 |
| 30-AWG White Wire, 6" Long | 30-W-50-060 | $1.38 |
| 30-AWG Red Wire, 6" Long | 30-R-50-060 | $1.38 |

® KYNAR-PENNWALT

MINIMUM ORDER $25.00, SHIPPING CHARGE $1.00, N.Y. CITY AND STATE RESIDENTS ADD TAX

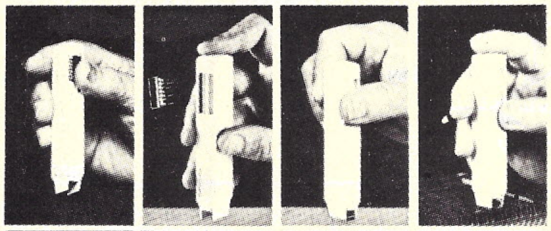
**OK MACHINE & TOOL CORPORATION**

3455 Conner St., Bronx, N.Y. 10475 ■ (212) 994-6600 ■ Telex 125091

OK®

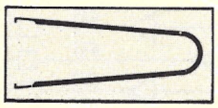
05

## DIP/IC INSERTION TOOL WITH PIN STRAIGHTENER



STRAIGHTEN PINS | RELEASE | PICK-UP | INSERT

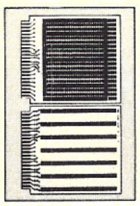
| | | |
|---|---|---|
| 14-16 Pin Dip IC Inserter | INS-1416 | $3.49 |



## DIP/IC EXTRACTOR TOOL

| | | |
|---|---|---|
| Extractor Tool | EX-1 | $1.49 |

## P.C. BOARD

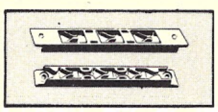


The 4 x 4.5 x 1/16 inch board is made of glass coated EPOXY Laminate and features solder coated 1 oz. copper pads. The board has provision for a 22/44 two sided edge connector, with contacts on standard .156 spacing. Edge contacts are non-dedicated for maximum flexibility.

The board contains a matrix of .040 in. diameter holes on .100 inch centers. The component side contains 76 two-hole pads that can accommodate any DIP size from 6-40 pins, as well as discrete components. Typical density is 18 of 14-Pin or 16-Pin DIP's. Components may be soldered directly to the board or intermediate sockets may be used for soldering or wire-wrapping.

Two independent bus systems are provided for voltage and ground on both sides of the board. In addition, the component side contains 14 individual busses running the full length of the board for complete wiring flexibility. These busses enable access from edge contacts to distant components. These busses can also serve to augment the voltage or ground busses, and may be cut to length for particular applications.

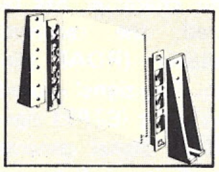
| | | |
|---|---|---|
| Hobby Board | H-PCB-1 | $4.99 |



## PC CARD GUIDES

| | | |
|---|---|---|
| Card Guides | TR-1 | $1.89 |

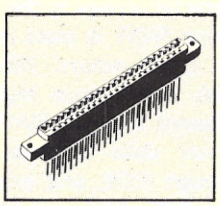
QUANTITY – ONE PAIR (2 pcs.)



## PC CARD GUIDES & BRACKETS

| | | |
|---|---|---|
| Guides & Brackets | TRS-2 | $3.79 |

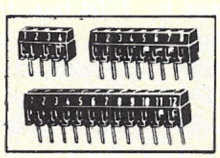
QUANTITY – ONE SET (4 pcs.)



## PC EDGE CONNECTOR

44 Pin, dual read out, .156″ (3,96 mm) Contact Spacing, .025″ (0,63 mm) square wire-wrapping pins.

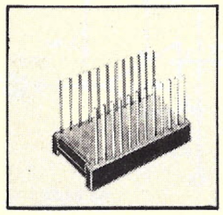
| | | |
|---|---|---|
| P.C.. Edge Connector | CON-1 | $3.49 |



## P.C.B. TERMINAL STRIPS

The TS strips provide positive screw activated clamping action, accommodate wire sizes 14-30 AWG (1, 8-0, 25mm). Pins are solder plated copper. .042 inch (1mm) diameter, on .200 incn (5mm) centers.

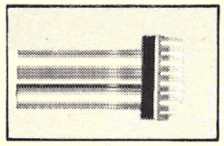
| | | |
|---|---|---|
| 4-Pole | TS- 4 | $1.39 |
| 8-Pole | TS- 8 | $1.89 |
| 12-Pole | TS-12 | $2.59 |



## DIP SOCKET

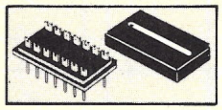
Dual-in-line package, 3 level wire-wrapping, phosphor bronze contact, gold plated pins .025 (0,63mm) sq., .100 (2,54mm) center spacing.

| | | |
|---|---|---|
| 14 Pin Dip Socket | 14 Dip | $0.79 |
| 16 Pin Dip Socket | 16 Dip | $0.89 |



## RIBBON CABLE ASSEMBLY SINGLE ENDED

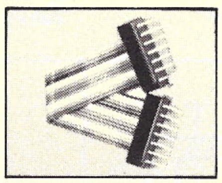
| | | |
|---|---|---|
| With 14 Pin Dip Plug 24″ Long (609mm) | SE14-24 | $3.55 |
| With 16 Pin Dip Plug 24″ Long (609mm) | SE16-24 | $3.75 |



## DIP PLUG WITH COVER FOR USE WITH RIBBON CABLE

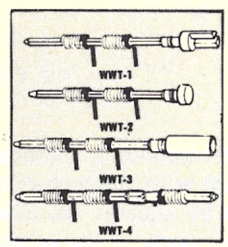
| | | |
|---|---|---|
| 14 Pin Plug & Cover | 14-PLG | $1.45 |
| 16 Pin Plug & Cover | 16-PLG | $1.59 |

QUANTITY: 2 PLUGS, 2 COVERS



## RIBBON CABLE ASSEMBLY DOUBLE ENDED

| | | |
|---|---|---|
| With 14 Pin Dip Plug - 2″ Long | DE 14-2 | $3.75 |
| With 14 Pin Dip Plug - 4″ Long | DE 14-4 | $3.85 |
| With 14 Pin Dip Plug - 8″ Long | DE 14-8 | $3.95 |
| With 16 Pin Dip Plug - 2″ Long | DE 16-2 | $4.15 |
| With 16 Pin Dip Plug - 4″ Long | DE 16-4 | $4.25 |
| With 16 Pin Dip Plug - 8″ Long | DE 16-8 | $4.35 |

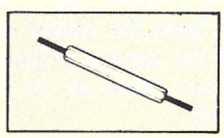



## TERMINALS

- .025 (0,63mm) Square Post
- 3 Level Wire-Wrapping
- Gold Plated

| | | |
|---|---|---|
| Slotted Terminal | WWT-1 | $2.98 |
| Single Sided Terminal | WWT-2 | $2.98 |
| IC Socket Terminal | WWT-3 | $3.98 |
| Double Sided Terminal | WWT-4 | $1.98 |

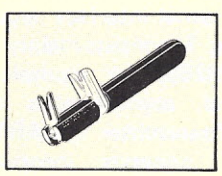
25 PER PACKAGE



## TERMINAL INSERTING TOOL

For inserting WWT-1, WWT-2, WWT-3, and WWT-4 Terminals into .040 (1,01mm) Dia. Holes.

| | |
|---|---|
| INS-1 | $2.49 |



## WIRE CUT AND STRIP TOOL

Easy to operate . . . place wires (up to 4) in stripping slot with ends extending beyond cutter blades . . . press tool and pull . . . wire is cut and stripped to proper "wire-wrapping" length. The hardened steel cutting blades and sturdy construction of the tool insure long life.

Strip length easily adjustable for your applications.

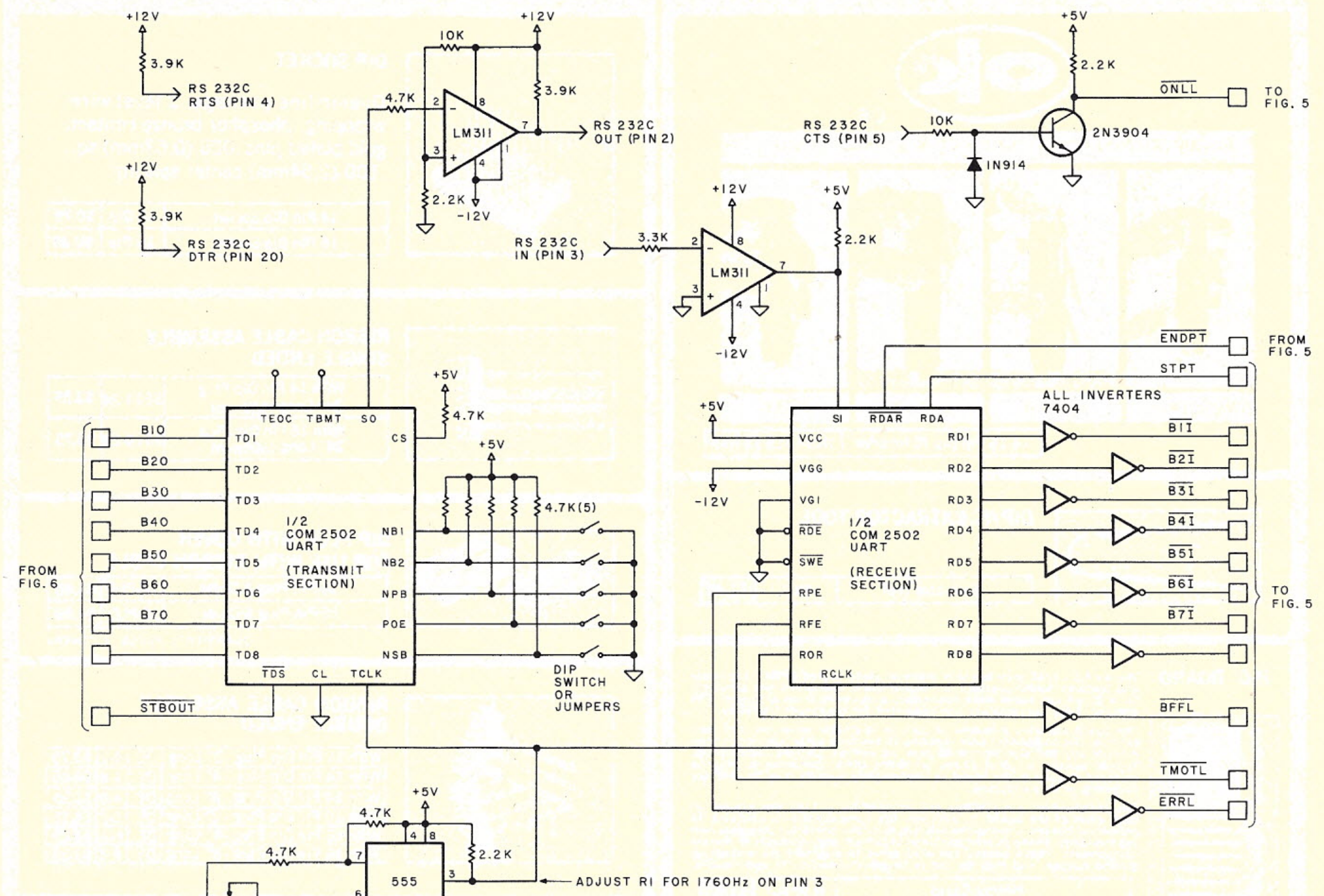
| DESCRIPTION | MODEL NUMBER | ADJUSTABLE "SHINER" LENGTH OF STRIPPED WIRE INCHES | TO | INCHES | Price |
|---|---|---|---|---|---|
| 24 ga. Wire Cut and Strip Tool | ST-100-24 | 1 5/16″ | — | 1 9/16″ | $ 8.75 |
| 26 ga. Wire Cut and Strip Tool | ST-100-26 | 1 3/16″ | — | 1 11/16″ | $ 8.75 |
| 26 ga. Wire Cut and Strip Tool | ST-100-26-875 | 7/8″ | — | 1 1/8″ | $ 8.75 |
| 28 ga. Wire Cut and Strip Tool | ST-100-28 | 7/8″ | — | 1 1/8″ | $11.50 |
| 30 ga. Wire Cut and Strip Tool | ST-100-30 | 7/8″ | — | 1 1/8″ | $11.50 |

THE ABOVE LIST OF CUT AND STRIP TOOLS ARE NOT APPLICABLE FOR MYLENE OR TEFLON INSULATION

MINIMUM ORDER $25.00, SHIPPING CHARGE $1.00, N.Y. CITY AND STATE RESIDENTS ADD TAX

**OK MACHINE & TOOL CORPORATION**

3455 Conner St., Bronx, N.Y. 10475 ■ (212) 994-6600 ■ Telex 125091




Note: Pin numbers refer to RS 232D connector numbers.

*Fig. 7. Serial interface.*

mitted to the computer, but not printed. The computer sends back, or echos, the character, and it is only then that it is printed.

The seven ASCII switches in the encoder matrix are buffered and inverted by a 7404 hex inverter and one section of a 7400 NAND gate. Therefore, data on the B10 through B70 lines is positive true. For example, the ASCII code for 4 is binary 00 110 100, and B30, B50 and B60 will be high while B10, B20, B40 and B70 are low. Note – the gates that generate B10 and B40 have two inputs isolated by 1N34 diodes. The second inputs needed to generate the proper ASCII codes when the Control Case switch is depressed.

Once again, the numbers in the circles indicate board edge connections that are made to wires taken from the main terminal block pin with the corresponding number. The card edge connectors indicated by squares are wired to other parts of the control circuits or to a microcomputer's input port. All outputs from the output section of the parallel interface circuit can drive up to ten TTL loads.

**Serial Interface**

The serial interface circuit (see Fig. 7) consists mainly of a COM2502 or equivalent universal asynchronous receiver/transmitter (UART), which converts incoming serial data to parallel data. Parallel ASCII data generated by the terminal is converted to serial data by this circuit before being transmitted. DIP switches or jumpers are used to set up the options for UART operation[4]. The 555 timer supplies the 1760 Hz clock signal[5] required for the UART to transmit and receive data at 110 BPS (or 10 cps).

The two LM311 comparators are used to convert the TTL-level signals required by the UART to ±12 V signals corresponding to the RS 232 standard. The request to send (RTS) and data terminal ready (DTR) lines of the RS 232C interface are held high by wiring them through 3.9k Ohm resistors to +12 V. The clear to send (CTS) signal is buffered and inverted by the 2N3904 transistor and is used to activate the On Line light.

Data on the B10 through B70 lines is strobed into the transmitter section of the UART by a negative-going pulse, $\overline{\text{STBOUT}}$, which is generated by Fig. 6 each time a key is pressed. The output data appears at the serial output (SO) pin of the UART at a rate of 1/16 the rate of the clock signal applied at TCLK.

Received serial data is applied to the UART's serial input (SI) pin. When a complete character has been received, the received data available (RDA) pin goes high. This signal is used as the start print (STPT) signal. The $\overline{\text{ENDPT}}$ signal generated by Fig. 5 is used to reset the UART's RDA signal. If a character arrives in the UART before RDA is reset an overrun error occurs. When the RDA signal goes high, this indicates that the UART's received data outputs RD1 through RD8, and error outputs RPE, RFE and ROR are valid. These signals are inverted by 7404 hex inverters before being sent to the parallel input circuits of Fig. 5. The outputs of the UART are designed to drive only one TTL load. Since some of the received data bits will drive the shift lock and unlock decoder as well as the circuits in Fig. 5, it is necessary to

| | | |
|---|---|---|
| $C_1,C_2$ | - | 2000 uF/30 V |
| $C_3,C_4$ | - | 1000 uF/35 V |
| $C_5,C_7$ | - | 10 uF/25 V |
| $C_6,C_8,C_{14}$ | - | 0.1 uF/50 V |
| $C_9,C_{10}$ | - | 50 uF/25 V |
| $C_{11},C_{12}$ | - | 68 uF/25 V |
| $C_{13}$ | - | 0.22 uF/50 V |
| $C_{15}$ | - | 10 uF/15 V |
| $C_{16}$ | - | 0.01 uF/50 V |
| $D_1$-$D_4$ | - | 6 Amp 100 PIV bridge or four 3 Amp 100 PIV rectifiers |
| $D_5$-$D_8$ | - | 1N4001 (optional) |
| $Q_1,Q_5$ | - | 2N3904 or similar |
| $Q_2,Q_6$ | - | 2N3906 or similar |
| $Q_3$ | - | MJE3055 on heat sink |
| $Q_4$ | - | 2N5194 on heat sink |
| $R_1,R_2$ | - | 120 $\Omega$ 1/2 W |
| $R_3,R_4$ | - | 470 $\Omega$ 1/2 W |
| $R_5,R_6$ | - | 0.3 $\Omega$ 2 W |
| $T_1$ | - | 25.2 VCT, 2 Amp Transformer |
| $U_1$ | - | LM340T-5 on heat sink |
| $Z_1$-$Z_4$ | - | 12 V, 1 W zeners – note $Z_3$ and $Z_4$ can be 13.5 V zeners and $D_5$-$D_8$ can then be eliminated. |

Note: Board allows for $R_1,Z_1$ and $R_2$, $Z_2$ to be replaced by a 78L12 and a 79L12 regulator, respectively.

*Table 5. Component list for 9350-2 terminal power supply.*

buffer the UART's output lines with the 7404s.

The Buffer Overflow light is activated by the UART's receiver overrun (ROR) signal. Similarly, receiver parity error (RPE) activates the Error light and the receiver framing error (RFE) signal turns on the Time Out light.

The bit 8 input (TD8) and output (RD8) pins of the UART are not used in this design.

**Shift Decode Circuits**

The shift decode circuit is shown in Fig. 8. The SHFTUP signal is generated for the uppercase alpha characters by detecting when the ASCII inputs bit B7I is high and B6I is low. Generation of the SHFTUP signal for the numeric characters is more complex because the keyboard-printer reverses some of uppercase and lowercase numeric characters as explained previously. This reversal occurs when both B3I and B4I are high. Therefore, these bits are NANDed together and Exclusive ORed with B51 and the result ANDed to B71 and B61 to generate the correct SHFTUP signal for the numeric characters.

Similarly the SHFTDN signal is generated for the lowercase alpha characters when B7I and B6I are both high. The SHFTDN signal for the numeric characters is generated by ANDing B31 and B4I together, Exclusive ORing this with B5I and ANDing the result with $\overline{\text{B7I}}$ and B6I.

All this sounds complex but may be understood by studying a table of the ASCII code set[2] and Fig. 8.

**Construction Notes**

The circuits shown in Figs. 4 through 8 are the starting point for the modification of the terminal. You can choose to build the circuits as designed or use only those portions that you need. My power supply is built on two printed circuit boards, while my control logic circuits are constructed on a wire-wrap socket board that has 21 14-pin sockets, seven 16-pin sockets, one 40-pin socket plus room for discrete components. If there is enough interest in these modifications to the Burroughs 9350-2, I'd consider laying out printed circuit cards for the control electronics. Write and let me know if you'd be interested. (Send an SASE.)

These construction notes are not intended to be step-by-step instructions for rebuilding the terminal. A job of this magnitude should not be attempted by someone without previous electronic design and construction experience. The notes are intended to serve as a guide to an experienced builder.

I will start by assuming that you have completed part 1 of the section, "Steps to Rebuild the Terminal," i.e., all pins have been removed from the main connector block and labeled appropriately.

1. Remove the plastic connector block and the 117 V ac connector from the aluminum mounting bracket and throw the connectors into your junk box for a future project. The aluminum mounting bracket will be used to mount the power supply rectifier and filter PC board, so don't lose it.

2. Mount the power transformer on the side frame close to the bottom of the terminal in the corner where the connector block was located. There is one hole available, so only one mounting hole need be drilled in the side frame.

3. Get the mounting bracket from step 1 and cut off the mounting tab that used to

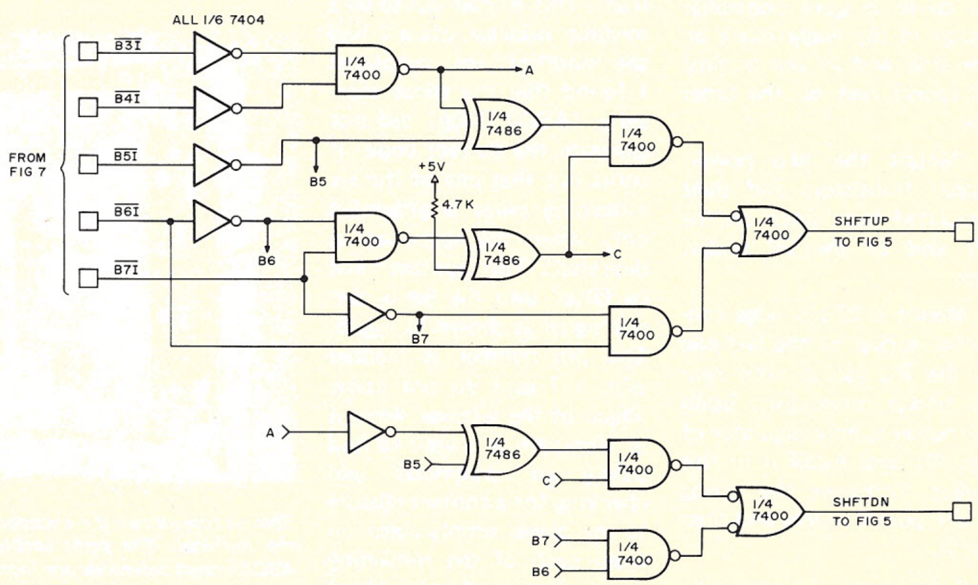



*Fig. 8. Shift decode logic.*

connect the bracket to the back of the terminal. Now install the mounting bracket back in the terminal, but upside down from its previous position. It should now be located directly above the power transformer.

4. Build the rectifier-filter PC board of Fig. 4a and install it on the mounting bracket, component-side down. Depending on the size of the filter capacitors, it may be necessary to file away a portion of the mounting bracket to obtain clearance. If your capacitors won't fit on the PC board, you can mount them in the extreme-back right-hand corner of the terminal, but you will have to devise your own mounting method and then wire them to the regulator-filter PC card.

5. Move the two sockets for the driver cards that come with the terminal to the first and third positions on the right-hand end of the PC socket rack at the back of the terminal.

6. Modify the PC socket rack to hold standard 22-pin .156-inch connectors. This can be done by using a ½ x ½ x 8¼ inch piece of aluminum angle stock. Bolt it to the PC socket rack at the left end of the terminal and at the small lip in the center of the terminal. Drill as many mounting holes as you have PC cards in your controller design in the angle stock on one side and in the original PC socket rack on the other side.

7. Mount the two power-supply transistors and their associated heat sinks on the left end of the PC socket rack.

8. Mount a 22-pin edge connector socket at the left end of the PC socket rack near the power transistors. Build the power-supply regulator of Fig. 4b and install it in this socket. Complete the wiring of the power supply and then test it.

9. Connect the wires from the main connector block to the edge connector (or connectors) that your control logic boards will plug into.

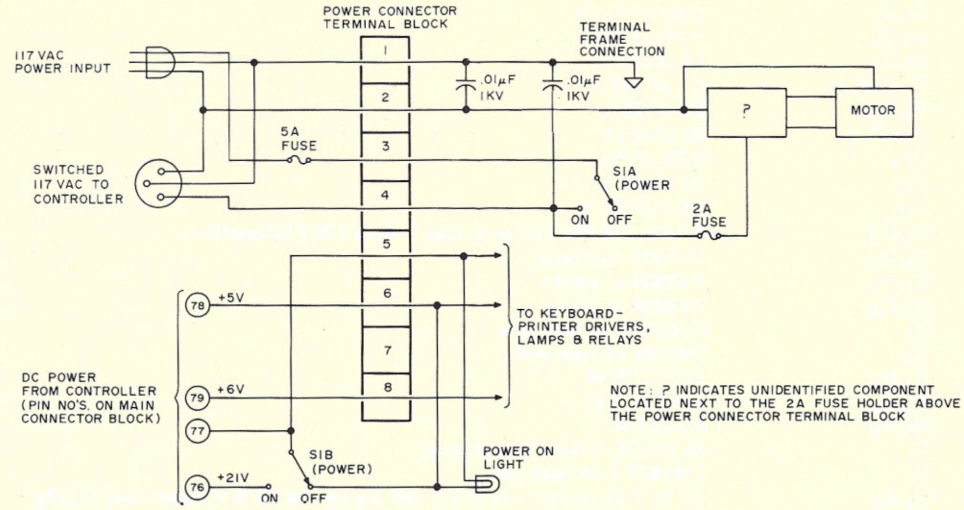
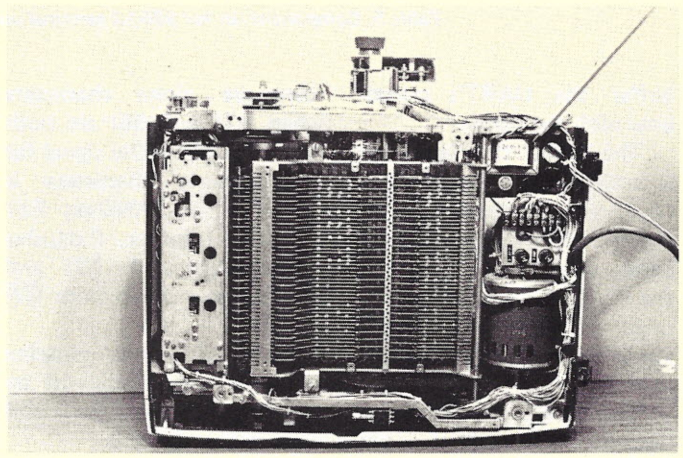



*Fig. 9. Original power-distribution wiring scheme.*

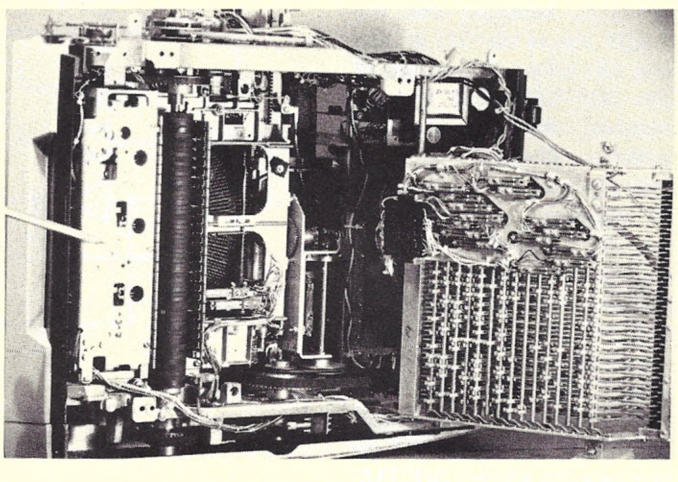
10. Construct and test the control logic circuits of Figs. 5 through 8.

At this point, if Murphy hasn't played too many tricks on you, your terminal should be ready to type its first listing.

In my case, Murphy did play a trick. There are half a dozen wires taken from the main connector block that I was unable to identify. However, my initial tests indicated that I did not require any of them. Therefore, I removed the markings that indicated which pin in the main connector block the wires came from. This turned out to be a mistake because when I had the modifications completed I found that the space character (ASCII 040$_8$) did not generate the correct code. It turns out that one of the six remaining wires is grounded only when the space bar is depressed. This contact must be ORed into the bit 6 output signal as shown in Fig. 6. The pin number is marked with a ? as I do not know which of the leftover wires it is. However, it is easy to find using an ohmmeter and checking for a contact closure to the power supply common from each of the remaining wires when (and only when) the space bar is depressed.



*View of the bottom of the keyboard-printer with cover removed. The bars in the center of the picture are part of the encoder matrix. The pointer indicates the transformer and filter capacitors, which were added by the author. They replaced the main connector block, which was removed from this area when the terminal was modified. The power connector block (Fig. 9.) is located below the transformer. Pin 1 of this block is to the extreme right.*



*This picture shows the encoder matrix swung out from the bottom of the terminal. The print enable, shift lock and unlock and the seven ASCII-coded solenoids are located behind the metal plate, as indicated by the pointer. Do not remove this plate as it is very hard to align it properly (voice of experience speaking).*

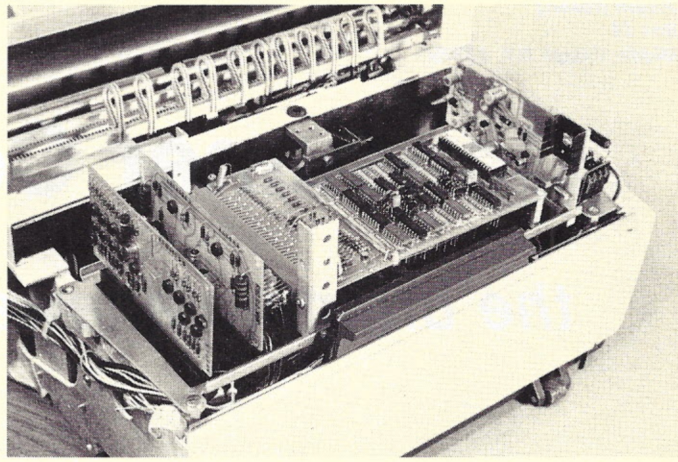
ABCDEFGHIJKLMNOPQRSTUVWXYZ

ABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890:-|{;},¯./

!"#$%&'()*=\[+]<^>?

*Fig. 10. Burroughs 9350-2 character set.*



*Here, all boards are installed to complete the reconstruction of the terminal. The back cover fits over these circuits in its original position. Note that the rectifier-filter PC board is installed below the two driver cards at the left of the picture and is, therefore, not visible.*

Anyone completing the modifications described in this article should keep track of the main connector-block number of the space function and then inform other *Kilobaud* readers of the correct pin number.

**Conclusions**

Because the Friden keyboard-printer is an ASCII-encoded device, it is easy to design simple control electronics as described in this article. It is much easier to understand the operation of the keyboard-printer than of the original controller. Even if your Burroughs 9350-2 communications terminal is working properly, you may want to consider using the circuits presented in this article for two reasons: 1. with the new circuits the terminal is software and hardware compatible with a Teletype or other terminal; 2. the new circuits are much simpler and are easier to troubleshoot. All the parts used are available from advertisers in this magazine. The old controller, on the other hand, uses obsolete ICs that, as far as I know, are not available anywhere. If a problem ever developed in the controller, it would be virtually impossible to fix — assuming that it could be found in the first place. ■


**References:**

1. Dan Stogdill, "Getting by the Friden 8800 Communications Gap — interface made easy," *73 Magazine*, December 1976.
2. Carl Helmers, "Deciphering Mystery Keyboards," *Byte*, September 1975. Page 65 has a table of the 7-bit ASCII code.
3. Gary Liming, "Watts Inside a Power Supply," *Byte* January 1977.
4. Don Lancaster, "Serial Interface," *Byte*, September 1975. Also COM2502 data sheet.
5. Jay A. Cotton, "Interface an ASCII Keyboard to a 60 mA TTY Loop," *Byte*, April 1976.

*Joseph Roehrig*
*Box 74*
*Middle Village NY 11379*

# Consumer Computer, Inc.

## *the ultimate answer to bill collectors!*

```
                              LIST

5 DIM B$(20),D$(20)
10 INPUT"DATE ? ",A$
20 INPUT"NAME ? ",B$
25 INPUT"ACCOUNT ? ",Z
30 INPUT"DATE OF LAST BILL ? ",C$
40 INPUT"AMOUNT & DEDUCTION ?",A,B
45 INPUT"DISPUTED FINANCE CHARGE ? ",C
50 INPUT"FIRM ? ",D$
52 INPUT"WHAT NUMBER FORM LETTER ? ",E$
90 D=25\E=A-B-C-D
94 INPUT"SET YOUR PAPER TURN ON THE PRINTER AND ENTER BLANK ",M$ '$
96 !" "\!" "\!" "
100 !"                                                     ",A$
101 !"                                                   ACC.",Z
102 !"PRESIDENT"
104 !D$
106 !" "
108 !"DEAR SIR:"\!" "
110 !"     MY CLIENT ",B$,"              ", HAS WRITTEN"
112 !"TO YOU ON NUMEROUS OCCASIONS REGARDING THE BILLS YOU"
114 !"ISSUE.  TO DATE, ",B$,"               " HAS ONLY RECEIVED"
116 !"COMPUTERIZED FORM LETTERS FROM YOU IN RETURN.  THIS HAS"
118 !"MADE IT IMPOSSIBLE TO SETTLE THE ACCOUNT.  IN ORDER TO"
120 !"FACILITATE THE SETTLEMENT CONSUMER COMPUTER WAS RETAINED"
122 !"BY ",B$,"."\!" "
124 !"     THIS IS THE ",E$," TIME I HAVE BEEN FORCED TO"
126 !"CORRESPOND WITH YOU AND I WILL CONTINUE TO REPLY TO ALL"
128 !"OF YOUR FORM LETTERS."\!" "
130 !"     THE SETTLEMENT THAT MY CLIENT AND I HAVE WORKED OUT IS"
132 !"AS FOLLOWS:"
134 !"            YOUR ",C$," BILL     ",%$10F2,A
136 !"            LESS DISPUTED ITEM    ",%10F2,B
138 !"            LESS FINANCE CHARGE   ",%10F2,C
139 !"            LESS COMPUTER CHARGE  ",%10F2,D\!" "
140 !"            BALANCE               ",%$10F2,E\!" "
142 !"     PLEASE FORWARD A WRITTEN (NON-FORM LETTER) ACCEPTANCE"
144 !"OF THIS OFFER TO MY CLIENT OR TO CONSUMER COMPUTER."
146 !" "
148 !"                                        THANK YOU"\!" "
150 !"                                           X   X"
152 !"                                            X X"
154 !"                                             X"
156 !"                                            X X"
158 !"                                           X   X"\!" "
160 !"                                     CONSUMER COMPUTER"
162 !"                                     POST OFFICE BOX 74"
164 !"                                     MIDDLE VILLAGE, NY 11379"
166 !"CC: BETTER BUSINESS BUREAU"
READY
```

*Fig. 2. Program listing.*

Have you ever had a problem with a credit-card company's billing computer? Regardless of how you attempt to solve the problem, the computer keeps sending you form letters and continues to add finance charge upon finance charge. Up to this point, the poor consumer had no recourse but to spend his valuable time responding to the form letters in an attempt to solve the problem.

Now, however, the microprocessor and the personal computer have come of age. The consumer lucky enough to have his own micro has a friend capable of doing battle with the giant company's oversized computer. The program contained in this article is relatively simple, but it can save the user much of the time normally spent answering form letters.


The microprocessor asks the beleaguered consumer eight easy questions that can be answered in less than a minute. Armed with this information, the micro dashes off to the aid of the consumer and generates a *consumer* form letter.

Fig. 1 shows the eight easy-to-answer questions and the form letter generated. As you can see, there is even a line in the letter to tell the company how many form letters were already sent. A giant-killing "CC: Better Business Bureau" is also included.

I'm using this letter for the first time and sending it to one of the largest credit-card companies. As you can see, I have a post-office box to lead the credit-card company into believing that a third party (Consumer Computer) is involved. I doubt that the $25 Consumer Computer fee will be taken off my bill, but it's worth a try. My Consumer Computer is the equal of the credit-card company's John Q. Cash, Manager of Collection.

This program took me about 20 minutes to write, and now I no longer need to answer form letters. It gave me some satisfaction to answer in this manner, rather than being frustrated answering oversized computers.

Fig. 2 is the program listing that you can easily edit to fit your needs. Let me know how you make out with your own Consumer Computer. ■

```
DATE ? 07/04/77
NAME ? JOSEPH J. ROEHRIG
ACCOUNT ? 333
DATE OF LAST BILL ? 06/15/77
AMOUNT & DEDUCTION ?103.22,9.89
DISPUTED FINANCE CHARGE ? .75
FIRM ? ANY FIRM
WHAT NUMBER FORM LETTER ? THIRD
SET YOUR PAPER TURN ON THE PRINTER AND ENTER BLANK
```

```
                                             07/04/77
                                            ACC. 333
PRESIDENT
ANY FIRM

DEAR SIR:

     MY CLIENT JOSEPH J. ROEHRIG, HAS WRITTEN
TO YOU ON NUMEROUS OCCASIONS REGARDING THE BILLS YOU
ISSUE.  TO DATE, JOSEPH J. ROEHRIG HAS ONLY RECEIVED
COMPUTERIZED FORM LETTERS FROM YOU IN RETURN.  THIS HAS
MADE IT IMPOSSIBLE TO SETTLE THE ACCOUNT.  IN ORDER TO
FACILITATE THE SETTLEMENT CONSUMER COMPUTER WAS RETAINED
BY JOSEPH J. ROEHRIG.

     THIS IS THE THIRD TIME I HAVE BEEN FORCED TO
CORRESPOND WITH YOU AND I WILL CONTINUE TO REPLY TO ALL
OF YOUR FORM LETTERS.

     THE SETTLEMENT THAT MY CLIENT AND I HAVE WORKED OUT IS
AS FOLLOWS:
                YOUR 06/15/77 BILL       $103.22
                LESS DISPUTED ITEM          9.89
                LESS FINANCE CHARGE          .75
                LESS COMPUTER CHARGE       25.00

                BALANCE                   $67.58

     PLEASE FORWARD A WRITTEN (NON-FORM LETTER) ACCEPTANCE
OF THIS OFFER TO MY CLIENT OR TO CONSUMER COMPUTER.

                                    THANK YOU

                                      X  X
                                       X X
                                        X
                                       X X
                                      X  X

                                CONSUMER COMPUTER
                                POST OFFICE BOX 74
                                MIDDLE VILLAGE, NY 11379

CC: BETTER BUSINESS BUREAU
```

*Fig. 1. Sample run.*

# Programmed Instruction Made Easy: Tiny PILOT

## *Part 1: language description*

*Allen S. Krieger*
*44 Webster Rd.*
*Lexington MA 02173*

PILOT is a nonmathematical computer language designed for dialogue-oriented, interactive, programmed instruction. It is widely used in educational applications ranging from the elementary grades through graduate school. PILOT should become part of the repertoire of any home computer hobbyist interested in computer-aided instruction (CAI) or man-machine interactive programming.

PILOT is simple; supposedly, first and second graders have been taught to write their own story-generating programs in PILOT. Because it is text oriented, PILOT can be used to teach nonmathematical, factual material in almost any subject from spelling to pharmacology. Most easy PILOT programs would be awkward or impossible to write in BASIC, a mathematically oriented computer language.

Because PILOT is simple in structure, it is relatively easy for a home computer hobbyist with no knowledge of computer science to write an interpreter for a "tiny" subset of PILOT. I know, because I did it. An *interpreter* is a computer code that reads and immediately executes programs written in a high-level, user-oriented computer language on a line-by-line basis. In contrast, a *compiler* translates programs written in a high-level language into machine-language code for later or repeated execution. Tiny PILOT interpreters written in assembly language usually occupy about 1K bytes of main memory in a home microcomputer. However, at least one Tiny PILOT interpreter has been written in Extended BASIC.

In the first part of this two-part series, I will introduce the Tiny PILOT language that I have been using to experiment with CAI on my microcomputer. I will describe the Tiny PILOT

| NAME | SYMBOL | FORMAT | |
|---|---|---|---|
| type | T | (%LABEL/) | T:text (/VRBLE/) (text)*CR* |
| ask | A | " | A:(/VRBLE/) *CR* |
| match | M | " | M:/MATCHSTRING/ (,/MATCHSTRING/)*CR* |
| yes | Y | " | – Y:x . . . x*CR* |
| no | N | " | – N:x . . . x*CR* |
| jump | J | " | J:/LABEL/*CR* |
| use | U | " | U:/LABEL/*CR* |
| return | R | " | R:*CR* |
| end | E | " | E:*CR* |
| zero | Z | " | Z:n*CR* |
| bump | B | " | B:n*CR* |
| examine | X | " | X:n = or < or > ccc*CR* |
| clear | C | " | C:*CR* |
| ignore | I | " | I:text*CR* |

**DEFINITIONS**

| | |
|---|---|
| (. . .) | Anything within parentheses is optional. |
| %LABEL/ | A statement label name of 1 to 5 characters preceded by %, and followed by /. |
| /VRBLE/ | A variable name of 1 to 5 characters preceded and followed by slashes. |
| text | Any ASCII character string that does not include a colon or a slash. |
| /MATCHSTRING/ | An ASCII character string of one to 15 characters preceded and followed by slashes. |
| -:x . . . x*CR* | Any Tiny PILOT statement (for use with Y or N). |
| n | I, J, K or L (in Counting Instruction statements). |
| ccc | Any positive, decimal integer constant between 1 and 255 (in the X statement). |

*Table 1. Tiny PILOT instructions.*

instructions (all 14 of them) and show what they can do. In the second part, I will describe my Tiny PILOT interpreter in enough detail so that anyone who knows even minimally how to program will be able to reproduce it.

**The Tiny PILOT Language**

PILOT was originally designed with simplicity of programming as a primary objective. The idea was to allow classroom teachers and ordinary students who are unsophisticated about computers and computing to write their own software for their own projects. A teacher should be able to dash off a PILOT program as fast as he or she would write a homework problem set or a workbook drill. Kids should be able to display their mastery of the subject matter by writing a program to teach it to someone else. In either case, the educational process is not advanced if the mechanics of the process are unnecessarily complex.

Simplicity is even more essential for PILOT in a home environment. Most of us will have a hard enough time figuring out what our PILOT programs should contain to help our own kids, without having to worry about the subtleties of a complicated language while we write the programs. Therefore, the author of a Tiny PILOT interpreter must be sure that in removing features of the parent language in order to fit it into the memory of a small home computer, he or she is making things easier, not harder, for the ultimate user. In my family, the ultimate user is seven years old.

That fact inspired several features of the Tiny PILOT design presented here. Tiny PILOT has no error messages—no one should ever feel put down by a computer. Tiny PILOT has no line numbers; statement labels are much easier to remember. Tiny PILOT allows variable names of up to five letters. That's a lot less confusing than single-letter names.

A Tiny PILOT program consists of *statements*, each ending with a carriage return. Each statement may have a label (optional), an *instruction* (mandatory), a colon (mandatory), an *operand field* (mandatory for some instructions, optional for others and ignored by a few) and a carriage return. The operand field of the statement is the text between the colon and the carriage return.

Table 1 lists the instructions in my Tiny PILOT (called KTP). Each instruction in KTP is represented by a single letter. There are 14 KTP instructions, but only two comprise the majority of most Tiny PILOT programs. These are the dialogue instructions, T and A.

**Dialogue Instructions**

T (for type) displays the text in the operand field of the instruction that is on the output terminal. The output terminal might be either a TV monitor or a Teletype. Program 1A is an example of the use of a T statement. Everything between the colon and the carriage return is typed, regardless of its length. If the last line of the text of the T statement is less than 32 characters long, it is padded with blanks. Should the text of the T statement contain a variable name (two slashes separated by one to five ASCII characters), the current text in the variable storage area with that name is printed. If no variable with that name can be found, KTP just prints the rest of the operand field of the T statement.

Program 1B is an example of the use of a variable in a T statement. Presumably, the contents of the variable, /NAME/, have been defined by a previous A statement (see below). For this example, assume it contains the name Billy.

In KTP, up to eight variables may be defined. Each variable may contain up to 63 characters of text plus a carriage return. This implies that 512 bytes of memory are used for variable storage. You may wish to allocate more storage by allowing more variable names. So far, I have not found these limitations restrictive.

The A (for Ask) statement accepts input from a keyboard and stores it temporarily in an input buffer. Up to 63 characters plus a carriage return can be accepted. A question mark is sent to the output terminal as a prompt character, and the input is echoed on the display as it is entered. If the A statement includes a variable name in its operand field (between colon and carriage return), the contents of the input buffer are stored, tagged with that

*The program reads:*

```
T:Hello, I'm Daddy's computer.
T:Who are you?
```

*The output terminal displays:*

```
Hello, I'm Daddy's computer.
Who are you?
```

*Program 1A. The T statement.*

*The program reads:*

```
T:Hello, /NAME/. Would you like to help me write
T:a story?
```

*The output terminal displays:*

```
Hello, Billy. Would you like to help me write
a story?
```

*Program 1B. Variables in the T statement.*

*The program reads:*

```
T:Hello, I'm Daddy's computer.
T:Who are you
A:/NAME/
T:Hello,/NAME/. Would you like to help me write
T:a story
A:
T:That's nice.
```

*The output terminal displays:*

```
Hello, I'm Daddy's computer.
Who are you
?Billy
Hello, Billy. Would you like to help me write
a story
? No I hate you, you rotten dirty computer
That's nice.
```

*Program 2. Use of T and A together.*

```
T:Would you like to help me write a story
A:
M:/yes/,/ok/
TY:That's nice.
TN:That's too bad.
```

*The output terminal displays:*

```
Would you like to help me write a story
?yes
That's nice
```

or:

```
Would you like to help me write a story
?no
That's too bad.
```

*Program 3. The M, Y and N statements.*

variable name. If no variable name is included, the input will vanish when the next A statement is executed. Program 2 puts the T and A statements together.

#### Input Test Instructions

Program 2 shows why it is obvious that we need a way to test the input and control program flow accordingly. The input is tested by the M instruction. An M (match) statement compares the contents of the input buffer with one or more *matchstrings* in its operand field.

A matchstring is made up of two slashes separated by up to 15 ASCII characters. If the comparison is successful, a program flag is set and the program proceeds to the next statement. If the match fails, the flag is reset and the next matchstring, if any, is tried. If the M statement contains no matchstrings, the flag is reset to "no," and execution continues with the next statement.

Matchstrings may begin adjacent to one another in the operand field of the M statement. Since any characters between the matchstrings are ignored, I always separate matchstrings with commas for clarity.

In order to control program flow based on the results of the M statement, KTP contains two flag-test instructions, −Y and −N. The Y means execute the instruction represented by the dash if the match flag is set to "yes." If the match flag reads "no," skip to the next statement. The N instruction is exactly the reverse. Any Tiny PILOT instruction can be used in a Y or N statement. Y and N statements do not change the state of the match flag; therefore, they can be chained. Program 3 shows how to handle both Bad Billy and Sweet Sara.

```
. . .
M:/yes/,/ok/
JY:/STORY/
T:That's too bad.
J:/EXIT/
%STORY/T: That's nice.
        T: Should this story be about a boy or a girl
        . . .
%EXIT/T: Goodbye, Let's play together again sometime.
     E:
```

*The output terminal displays:*

```
        . . .
        ?yes
        That's nice.
        Should this story be about a boy or a girl
or:
        . . .
        ?no
        That's too bad.
        Goodbye, Let's play together again sometime.
```

*Program 4. The J and E statements.*

```
. . .
T: Hello,/NAME/. Would you like to help me write a story
U:/YORN/
JY:/STORY/
. . .
%YORN/A:
      M:/yes/,/ok/
      R:
```

*The output terminal displays:*

```
      Hello, Billy. Would you like to help me write a story
      ?ok
      That's nice.
      . . .
```

*Program 5. A Tiny PILOT subroutine.*

#### Program Branching Instructions

The J (jump) instruction is used for program branching. The operand field contains a label name that consists of one to five ASCII characters preceded and followed by slashes. Execution will continue with the statement preceded by the label named in the J statement. The label, however, begins with a percent sign instead of a slash. If the label cannot be found, KTP continues execution with the statement after the J statement. The J instruction can be used unconditionally, or it can be used with Y or N as the response to a test. Progam 4 shows another way to handle a question response.

Program 4 also introduces the E (for end) instruction, which terminates execution of a Tiny PILOT program. As implemented in KTP, an E statement causes the computer to go into a loop while the last display on the TV screen is read. Pressing any key on the keyboard then returns control to the operating system.

Tiny PILOT has subroutines, too. They are called by the U (for use) instruction. As in the J statement, the U statement causes a jump to the statement labeled with the character string set off by slashes in the operand field. The difference is that the U statement causes the interpreter to save the address of the calling statement. Upon encountering an R (for return) instruction in the subroutine, KTP jumps back to the statement in the program following the U statement. Program 5 shows a short subroutine that saves space in a program that asks for a lot of yes or no answers. It is used in the prologue of the story-generating program of the last few examples.

#### Counting Instructions

KTP also provides some limited, but powerful, capabilities for counting. KTP includes four one-byte counters—I, J, K and L. Three instructions allow the Tiny PILOT programmer to manipulate a counter named in the operand field of a statement. He or she can set a counter to zero with the Z instruction, increment a counter with the B (for bump) instruction or compare the current value of the counter to a programmer-specified, decimal integer constant with the X (examine) instruction. The X statement allows the programmer to specify whether he or she wants the match flag set to Y for a counter greater than, less than or equal to the constant.

The counter manipulation instructions make Tiny PILOT into a real tool for programmed instruction. With these instructions, you can keep track of right or wrong answers for several categories of questions. You can cause the program to branch to additional drill on a particular type of problem if the wrong answer count exceeds a predetermined value. You can jump to more advanced work if the number of correct answers to a series of questions is higher than some standard. The primary use of the counters, though, is to enable you to implement simple loops.

Program 6 shows the use of the Z, B and X instructions in a factual drill in elementary geography. Counter I is a loop-control register in this program, while counter J keeps track of correct answers. Notice that we must zero the counters before we use them.

The program types the question we wish to ask—in this case, the names of the six New England states. It then enters a loop at the statement labeled %ASKLP/. The A statement accepts the student's answer. The M statement then checks the answer against the six correct answers. Spelling counts. Unless the student's response contains a character string that exactly matches one of the six correct answers, the match flag will indicate "no." However, if our geography scholar has typed the correct answer, the BY statement adds one to his score in counter J.

Either the TY or TN statement is now typed to tell the student how he did. The pro-

*New!* **SELECTERM** SYSTEM 9710

# The SELECTRIC II* Printer you can TRUST

**BECAUSE . . .** It's brand new, and fully assembled and tested.

**BECAUSE . . .** After extensive engineering design and testing by Micro Computer Devices, IBM Corporation has approved the SELECTERM for use with your microcomputer, and provides you with their factory warranty and yearly service agreement for the typewriter. In addition, the electronics conversion portion is fully warranteed by Micro Computer Devices.

**BECAUSE . . .** You can connect the SELECTERM to your computer within minutes of taking it out of the carton.

IT'S THAT EASY! AND THAT RELIABLE!

**FEATURES**
- Complete ASCII character set in supplied element.
- Full upper, lower case alphanumeric characters.
- Tab Command, Index (vertical tab), Backspace, Bell—all under computer control.
- Parallel Interface, standard.

**ALL ELECTRONICS INCLUDED**
- Power supply, electronics and cable sets included to permit immediate connection to the parallel port of any computer, at standard TTL level.

**SOFTWARE**
- All necessary conversion software in PROM to handle ASCII input, directly.

**PRINTER or TYPEWRITER**
- May be used as a standard typewriter when not in use with your computer.

**OPTIONS**
- Dual Pitch, $100
- Correction Feature, $100
- Tractor Feed Platen, $250
- Noise Reduction Feature, $40

**AVAILABLE SOON**
- RS-232 Interface

**PRICE and DELIVERY**
- Assembled and tested, $1650
- Available ONLY from authorized dealers.
- Delivery 1 to 2 weeks from receipt of order.

SELECTERM

**micro computer devices inc.**
960 E. Orangethorpe, Bldg. F
Anaheim, California 92801
Telephone (714) 992-2270

*Registered trademark of IBM Corporation

M30

*"Innovators to the Microcomputer Industry"*

```
Z:I
Z:J
T:Name the six New England states
%ASKLP/A:
        M:/Maine/,/New Hampshire/,/Vermont/,/Massachusetts/,
            /Rhode Island/,/Connecticut/
        BY:J
        TY:Right!
        TN:No
        B:I
        X:I = 6
        TN: Name another one
        JN:/ASKLP/
X: J>4
TY: You really know your geography
TN: The six New England states are . . .
E:
```

*The output terminal displays:*

```
        Name the six New England states
        ?Main
        No
        Name another one
        ?Vermont
        Right!
        Name another one
        ?New Hampshire
        Right!
        Name another one
        ?Massachusets
        No
        Name another one
        ?New York
        No
        Name another one
        ?Rhode Island
        Right!
        The six New England states are Maine, New Hampshire, Vermont,
        Massachusetts, Connecticut, and Rhode Island.
```

*Program 6. The Z, B and X instructions.*

gram increments counter I to indicate that it has nearly completed another pass through the loop. The X statement examines the contents of counter I. If counter I is equal to six, the loop is done, so we want the match flag set to "yes." If counter I is less than six, the TN statement asks the student to name another New England state. The JN statement brings the program back to %ASKLP/ to process another answer.

When the loop is done, we'd like to know how well the student did. The X statement checks the J counter. If the student got five or six correct answers, the TY statement tells him the names of the New England states. The E statement gives the student a chance to read the last message. It will return control to the monitor as soon as any key of the keyboard is tapped.

This program would have been in serious trouble if we had failed to set the counters to zero at the start. The program checked the number of passes through the loop by testing whether counter I equaled six. If counter I had contained a number larger than six when we started, the loop would have repeated until the counter counted up to 255 and went through zero. It then would have repeated six times more. Suppose we change the counter test instruction to X:I>5? That almost works. If counter I starts out larger than six, the program goes through the loop exactly once. There's no substitute for initializing variables.

I've written Program 6 specifically to demonstrate the use of the Tiny PILOT counters, both for loop control (counter I) and for score keeping (counter J). It is not a very educational program. Knowing how to spell some complicated, mispronounced words of Algonquin Indian dialect is not a particularly useful skill after you leave the fifth grade. You should ask some questions about each state. Who can forget Maine lobster, Vermont maple sugar, Connecticut insurance companies or New Hampshire hobby-computer magazines? Also, the program has a minor bug that any ten-year-old is sure to find on his second or third run through the whole loop. Try typing in Maine six times in a row.

### Miscellaneous Instructions

There are two more instructions in KTP. The I (ignore) instruction is used to insert comments or remarks in a Tiny PILOT program. Anything in the operand field of an I statement is ignored by the KTP interpreter. Tiny PILOT is almost completely self-documenting because variable names and statement labels are long enough to be meaningful. It helps to write down the purpose of the counter registers, though, because they have one-letter names. I find it also helps to have a couple of lines describing the main points of the program at the start of a listing.

The C instruction clears the screen of a page-mode TV typewriter to prevent overwriting the display on the screen by typing too many lines. You can ignore this instruction if your output device is a Teletype or if your TV typewriter is equipped with a scrolling capability. Scrolling means that when you add a new line to a full screen, all the lines on the screen roll up by one line, the top line disappears and the new line appears at the bottom of the screen—as if you were reading a scroll.

Most TV typewriters don't have scrolling capability. If you try to add a new line to the bottom of a full screen, it appears at the top of the screen, replacing whatever was there before. This can be most confusing. The C instruction clears the screen. The C statement does not use the operand field; it is just C:*CR*. If you wish, you may include comments in the operand field. They will be ignored by the KTP interpreter.

If you use the C instruction, remember that you must leave the user time to read any text typed on the screen. If the program goes straight from a T statement to a C statement, the screen will be cleared a few milliseconds after the text is typed. There are two ways to provide reading time. The easy way is to precede a C statement by an A statement. The program then waits for input before clearing the screen. This could be awkward to program, however, so, as an alternative, you can write a subroutine that wastes time by incrementing the counter registers over and over many times. You'll have to experiment with your own computer to determine the correct timing constants. The length of the pause is an esthetic matter that depends very much on the

reading ability of your user.

### Putting It All Together

KTP requires a separate text editor in order to get Tiny PILOT programs into your computer. If your operating system already includes a text editor, use it. If it does not, you can buy, copy or write a text editor for use with Tiny PILOT. Text editors are not very hard to implement, and several good ones are available to the hobbyist market. In choosing one, remember that for Tiny PILOT, convenience and simplicity are much more important than elegant features. It doesn't make much sense to take a language that can be programmed by a seven-year-old and couple it to an editor designed for PhDs in computer science. Also, Tiny PILOT does not use line numbers. If your text editor requires them, you will lose about 15 percent of the Tiny PILOT text area in your memory.

On my system, I was able to allocate 6K bytes to the Tiny PILOT text area. If I were to write a Tiny PILOT program consisting of nothing but T statements, I could fill the screen 12 times. In practice, the elementary-school CAI programs that I write fill the screen a lot more often than that. I don't recall ever being limited by the size of my memory.

By now, I hope you agree with me that Tiny PILOT is absolutely essential for any home computer owner with school-age children; it might be a lot of fun even for people without kids. It could be just the application to impress non-technically oriented friends and relatives, or to help you brush up on a foreign language before a vacation trip. How do you get a Tiny PILOT interpreter for your computer?

### The Interpreter

Masochists who like hand-loading machine-language programs (the way I originally wrote KTP) will appreciate program listing #1. It is a hex dump of the KTP object code in Z-80 machine language. It is written for Digital Group computers with at least 10K of memory. The format of the listing consists of a line address, 16 bytes of memory and a checksum of the 16 bytes added together without carries. This allows you to check your hand-loading by writing a little program to compute and display the checksums before you try to run KTP. That idea was suggested by Brent Longtin, of Algorithmics Incorporated, who was kind enough to produce the memory dump with his Diablo printer.

If your Z-80 computer is not a Digital Group system, you will have to change some calls to system utility routines to the correct addresses for your operating system. The routine at

```
880  cd e6 00 cd  76 0b 21 00  10 01 a3 ba  cd 22 0b 38  c2
890  22 22 5d 0d  cd 2d 0b fe  e0 38 02 e6  df 21 d0 0d  8e
8a0  01 0e 00 ed  b1 20 15 7d  c6 0f 6f 5e  7d c6 10 6f  c3
8b0  56 eb e9 cd  a8 01 c7 cd  e6 00 18 0e  2a 5d 0d 06  da
8c0  20 23 cd 00  0b 38 08 cd  16 0b 2a 5d  0d 18 ba cd  7c
8d0  34 0b e5 c5  cd 49 0b 38  15 21 00 0e  78 a7 28 06  d3
8e0  11 40 00 19  10 fd c1 a7  cd 00 0b e1  18 d3 c1 e1  25
8f0  18 cf 11 60  0d 06 40 af  12 13 10 fc  3e bf cd fa  4f
900  00 06 3e 11  60 0d cd 03  06 12 fe 8d  28 08 13 10  88
910  f5 3e 8d 12  18 0c 78 d6  20 28 07 fa  1f 09 47 cd  c9
920  16 0b 2a 5d  0d 01 8d af  cd 22 0b da  ca 08 cd 34  99
930  0b cd 49 0b  30 2b 3a 5f  0d fe 08 d2  ca 08 21 a0  98
940  0d 47 a7 28  0b 7e 23 fe  af 20 fa 10  f8 3a 5f 0d  44
950  4f 3c 32 5f  0d c5 eb 21  57 0d 01 06  00 ed b0 c1  c3
960  41 21 00 0e  78 a7 28 06  11 40 00 19  10 fd eb 21  40
970  60 0d 01 40  00 ed b0 c3  ca 08 af 32  56 0d 2a 5d  ab
980  0d 22 54 0d  01 8d af 2a  54 0d cd 22  0b 38 26 cd  7d
990  b6 0b 22 54  0d 21 43 0d  46 eb 21 5f  0d cd 22 0b  6d
9a0  38 e2 22 41  0d cd 98 0b  38 da 28 05  2a 41 0d 18  c9
9b0  ec 21 56 0d  34 c3 ca 08  af 3c 21 56  0d be c2 ca  f2
9c0  08 2a 5d 0d  cd 2d 0b c3  94 08 af 18  ed cd de 0b  6a
9d0  da ca 08 2a  54 0d c3 89  08 3a 40 0d  a7 c2 ca 08  4d
9e0  2a 5d 0d 22  3e 0d cd de  0b da ca 08  21 40 0d 34  05
9f0  2a 54 0d c3  89 08 21 40  0d 7e a7 ca  ca 08 af 77  34
a00  2a 3e 0d c3  89 08 cd 07  0c 38 4f eb  34 18 4b cd  7f
a10  07 0c 38 46  af 12 18 42  af 32 56 0d  cd 07 0c 38  08
a20  39 ed 53 41  0d cd 1d 0c  38 30 fe bc  38 2c fe bf  00
a30  30 28 0e 00  fe bd 28 06  38 03 0c 18  01 0d cd 1d  a6
a40  0c cd 34 0c  38 14 2a 41  0d be 28 06  38 03 0c 18  28
a50  01 0d 79 a7  20 04 21 56  0d 34 c3 ca  08 fe fe fe  99
a60  78 f6 ea f4  74 7c fc 75  76 7e 5c f4  ff 7c f4 dc  3c
a70  fe fe ce fe  7c fe fc fe  fe de 7e 5a  da de de 7e  04
a80  f4 f0 7c 70  f4 5a f0 fe  f5 50 f8 f0  fc 70 70 5a  6f
a90  fe fe fa fc  fa 7a da 5c  5e de f6 db  de fa fe ee  6d
aa0  f4 72 54 5c  74 f0 de f4  74 76 fa fc  50 74 58 f8  40
ab0  fa fa de fa  76 fa 7a fe  da b6 7e 4e  de de fe ee  b8
ac0  f0 fa 70 34  74 52 50 70  74 50 d4 f2  50 f0 50 50  7e
ad0  5e 5e fe 5e  5a 5c fe de  be 5e 6b e8  f6 fe 5e 7e  e9
ae0  f8 74 58 f0  50 7c d0 f2  fc f0 58 50  f4 7c d8 7a  98
af0  ea ee da de  ae fa 7e de  da fe ff ce  fe fb 7e ca  7a
b00  11 af 8d 7e  ba 28 0e bb  28 0a cd fa  00 23 10 f3  95
b10  06 20 18 ef  37 c9 78 fe  20 c8 3e a0  cd fa 00 10  40
b20  f9 c9 23 7e  b9 28 04 b8  20 f8 c9 37  c9 2b 7e fe  88
b30  a0 28 fa c9  c5 01 af 05  11 57 0d 23  7e 12 b9 28  0e
b40  06 23 13 10  f7 79 12 c1  c9 3a 5f 0d  a7 28 25 01  f3
b50  af 00 11 a0  0d 21 57 0d  1a b9 28 07  be 20 07 13  ec
b60  23 18 f5 7e  b9 c8 04 3a  5f 0d b8 28  07 1a b9 13  a6
b70  28 e3 18 f9  37 c9 06 30  21 a0 0d af  77 23 10 fc  75
b80  21 00 0e 01  02 00 77 23  10 fc 0d 20  f9 21 5f 0d  8b
b90  77 32 56 0d  32 40 0d c9  c5 d5 06 af  1a b8 28 0b  a8
ba0  be 28 0f 7e  b9 28 07 a7  3c 18 04 af  18 01 37 d1  2a
bb0  c1 c9 13 23  18 e6 c5 01  af 0f 11 43  0d c3 3b 0b  ac
bc0  c5 01 af 05  18 f4 0e af  21 43 0d 11  57 0d 1a b9  fc
bd0  28 07 be 20  07 13 23 18  f5 7e b9 c8  37 c9 2a 5d  dd
be0  0d 01 8d af  cd 22 0b d8  cd 34 0b 21  ff 0f 22 54  cd
bf0  0d 2a 54 0d  01 a3 a5 cd  22 0b d8 22  54 0d cd c0  c3
c00  0b cd c6 0b  38 eb c9 2a  5d 0d cd 1d  0c d8 fe c9  be
c10  d8 fe cd 30  13 11 3a 0d  d6 c9 83 5f  c9 23 7e fe  27
c20  8d 28 05 fe  a0 28 f6 c9  37 c9 fe b0  d8 fe ba 3f  bc
c30  d8 e6 0f c9  cd 2a 0c d8  57 23 7e cd  2a 0c 38 0f  b3
c40  5f 7a 17 d8  17 d8 82 d8  17 d8 83 d8  57 18 ea 7a  2e
c50  a7 c9 b7 3e  3e bf b7 16  36 ff b7 36  3e bf b7 36  3b
dd0  c5 c3 c9 d4  c1 cd d9 ce  ca d5 d2 c2  da d8 00 00  3f
de0  b3 b7 ba bc  f2 7a b8 ca  cd d9 f6 06  0f 18 00 00  97
df0  08 08 08 08  08 09 09 09  09 09 09 0a  0a 0a 00 00  7c
```

*Listing 1.*

0E6H clears the screen and initializes the cursor at the upper-left corner of the display. The routine at 0FAH writes a single character from the accumulator into the next available location on the TV screen. The routine at 01A8H reads a single character from the keyboard into the accumulator. Even the users of Digital Group systems will have to duplicate the homebrew routine addressed at 0603H. This routine echoes a character entered from the keyboard on the display and leaves it in the accumulator.

As written, KTP loads from location 0880H to location 0A59H and from location 0B00H to 0C51H. The KTP command tables are loaded from 0DD0H to 0DFDH. The rest of page 0D is scratchpad storage. Variables are stored on pages 0EH and 0FH. The text area for Tiny PILOT programs (which must be accessible to your editor) runs from 1000H to the top of memory.

If you use an 8080, you can still use this code, but it will not be easy. You will need a manual of Z-80 op codes. When you find an unrecognizable instruction, you will have to look it up. Chances are it is a two-byte relative jump; you will have to insert a three-byte 8080 style absolute jump. You will also have to write small routines to emulate the Z-80 block-comparison and block-move instructions. If you have a macro-assembler, this will be simple. Just define macros for the Z-80 instructions CPIR and LDIR.

For people who like doing things the easy way, a Digital Group format cassette tape copy of KTP can be obtained for $15 from Computer Mart, Inc., 1097 Lexington St., Waltham MA 02154. The cassette includes a Tiny Text Editor that can remain resident along with KTP to simplify debugging Tiny PILOT programs. 8080 users with Teletypes can obtain a paper-tape copy of a more difficult subset of PILOT called PILOT-8080 from the National Library of Medicine, 8600 Rockville Pike, Bethesda MD 20014 (see reference 3).

I advise you to write your own Tiny PILOT interpreter. It is not a difficult task. You will learn the essence of interpreter operations without having to simultaneously handle the complications introduced by more complex high-level languages. You will understand your Tiny PILOT interpreter in a way that you can never know a program written by someone else. If you don't like a particular feature of the language, you can change it. Finally, if you are like me, it will give you enormous satisfaction to have written a sophisticated piece of systems software while still only a beginner—and that can be used by beginners. ■

---


1. "PILOT," *People's Computer Company,* Vol. 5, No. 3, Nov-Dec 1976, p. 10. (An introduction to PILOT programming in *Whole Earth Catalog* style. In general, *People's Computers*—the new name of *PCC*—has started to publish at least one Tiny PILOT program in each issue.)

2. Gregory Yob, "PILOT," *Creative Computing,* Vol. 3, No. 3, May-June 1977, p. 57. (A discussion of PILOT in the classroom context by the coordinator of the PILOT Information Exchange.)

3. John Starkweather, "Guide To 8080 PILOT, Version 1.1," *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia,* Vol. 2, No. 4, April 1977, p. 17. (A computer-generated description, specification and test program. Dr. Starkweather was the author of the original PILOT at the University of California, San Francisco, Medical Center. He used it to teach pharmacology to medical students.)

4. PILOT Information Exchange, c/o Gregory Yob, PO Box 354, Palo Alto CA 94301. (This is the users' group for PILOT. They maintain a library of PILOT programs and lists of known PILOT installations. They also publish a newsletter of interest to professional educators using PILOT in schools.)

# Protect Your Memory

## *against power failure*

*Charles R. Carpenter*
*2228 Montclair Pl.*
*Carrollton TX 75006*

Concerned about losing everything in your computer's memory during a power failure? I was, and here's a circuit to prevent memory loss when the lights go out.

Select a rechargeable battery to handle the load—a bank of Nicads, small-car or motorcycle battery—anything that meets your needs. Connect one LM 340-5 in the circuit for each 1.2 Amps of current used to run the circuits you're going to protect. Mount the regulators on a good heat sink.

The diodes connected to the output are used to control current flow.

In normal operation, the diodes act as a diode switch and are biased off by a slightly higher voltage from the computer. During a power failure, current is supplied from the batteries. One high-current diode can be used in the output, but if your memory can run (idle) on 4 volts, then use whatever you have. There is a forward voltage drop of about 1 volt across a silicon diode. (I used several in parallel to reduce the drop to about .5 volt—diodes don't reduce like resistors.)

Trickle charge the battery to keep it at maximum potential. Select a resistor that draws about one-half the trickle-charge current to load the output. The battery will stay charged and last longer if the current flow is continuous.

I use a Radio Shack 12 volt power supply for a charger. It works fine.■

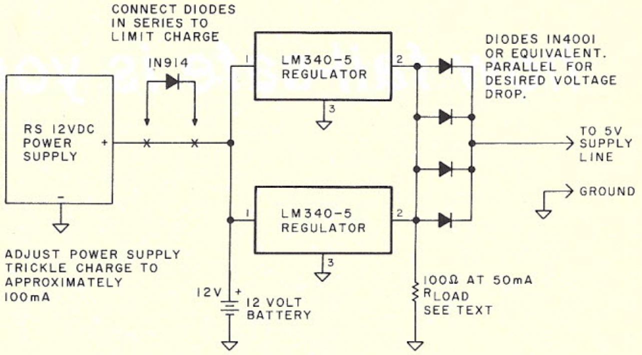



*Fig. 1. Memory saver schematic.*

# Backup Techniques

## *how fail-safe is your system?*

*William L. Colsher*
*2110 Hassell Rd. Apt. 308*
*Hoffman Estates IL 60195*

Let's start with a little story about a data processing installation and how it works. It's 5:30 pm or so, and a store owner drops off the day's receipts to be entered into the computer system he has contracted to use. A clerk takes the receipts to a place called Data Entry, where rows of people sit at keypunches (the machines that make the little holes in computer cards) turning receipts, orders or payroll time cards into a form the computer can understand. Another clerk takes the completed card deck to the computer room, where a third clerk adds some more cards that tell the computer to run, say, the inventory program rather than payroll or accounts receivable.

Next, the entire card deck is read into the computer, and when it has the time, it executes (an unfortunate term) the program. When the computer calls for the master file tape, an operator mounts it on a tape drive (a machine that will read the data stored on the tape). All goes well, and a new master file is written out by another tape drive as the old one is read. The old master tape is put in a stack of tapes that can be used again since they now have a new, updated version of the file. Another clerk looks over the printouts to verify that everything is in balance.

Since the program worked as expected, they throw out the cards that were used. Unfortunately, no one noticed that when the new master file tape was rewound, one of the two motors that turn the tape reels ran just a little faster than the other and the tape was stretched. Next day, the computer cannot read the tape. The previous master is gone, used by another program. What can we do? Somehow we'll have to recreate the master file, probably from an old listing somewhere, and then enter all the transactions that have occurred since that version of the file was current. All this involves a lot of work and will cost somebody a fortune.

### The Importance of Backups

Luckily, most companies that do data processing for others are more careful than this. The few that aren't don't last long. Rather than toss out the cards and reuse the tape, they save both for at least a week, and often much longer. Sometimes, too, the input card decks are copied onto tape, where they are much safer than they would be stacked in some file cabinet. For added safety, these backup tapes, as they are called, are usually kept in a vault separate from the one housing the current tapes.

Do I hear you saying that you don't need to do this? That you're careful with your tapes and nothing will happen to them? Whom the gods destroy, they first make mad. You *will* need those backups. Two weeks ago, we had two little cousins over for the holiday weekend. One of them, wishing to record some songs from my record collection, inquired about a spare tape. I directed him to the top shelf on the left of the computer. He sped unerringly to the shelf to the right of the computer, and quicker than you can say, "Surf's up!" my new, peaceful version of Star Trek was replaced by "Catch a Wave" and "I Get Around." Was three weeks of effort lost? Would the young cousin die a slow and horrible death? Nope. I went down to the bank the next day and picked up my backup tape from my safe deposit box, copied the program onto a new tape, and life was restored to normal.

By now you ought to have some idea of the importance of backing up your programs and data. Both are equally important because all the master files in the world won't do you a bit of good without a program to process them; and the best inventory system money can buy is so much used tape if the files can't be read.

### How and When to Back Up

Now that you have the whys and wherefores of backups fixed firmly in mind, the next question should be: "How and when do I back up my stuff, and how often?" The how of backups is simple in the extreme. All you need is a program to copy one tape onto another (or dump a disk to tape if you have floppies), and a comparison program to read the original and compare it to the copy. This assures you that the copy is completely identical to the original and that the tape has not been damaged. I've had some very expensive audio tapes right out of the box that were useless for my system, so you can see that the comparison is a good idea. In addition, a comparison routine is the only way to verify that a long complex program has been copied correctly. One technique, which I've found useful even on the big machines, is to put the tapes on different tape drives during the comparison. In other words, if the copy was created on

drive A and the master is being read on drive B, the comparison is run with the copy on drive B and the old master on drive A. This will insure that you are independent of the peculiarities of any given machine.

The question of when and how often to back up your files depends on how you run your business. A good general rule that provides maximum protection is to back up a given file immediately after running any program that changes that file. This can run into a lot of tapes and a fair amount of extra time, but what is your accounts receivable file worth? A top-quality *certified* (tested by the manufacturer) cassette tape with room for two or three full floppies only costs six or seven bucks. Were they lost, could you recover all your current files for six dollars? $60? $600? At all? If you spend a little extra time and money on backups, you can save yourself a lot of headaches later. You *will* use those backups, maybe as many as five or six times in the first couple of months you have your system.

Making frequent backups will help protect your business from financial disaster due to human error or some minor machine malfunction. By minor I mean dirty heads, stretched tape and the like. Speaking of this type of failure, almost nothing will ruin a tape faster than leaving it in a locked car all day. Even during the winter it can get awfully hot in a glove compartment or trunk. What will you do if your whole system blows up? Say your power supply fails or, heaven forbid, your office burns down, or someone steals your computer (pretty soon they will be as popular as stereos and color TVs). What happens then? If you had a good consultant or had read this article before you bought your system you would have a "backup site." This could be a number of different things, but the idea is to provide you with timely access to a system that has at a minimum, capabilities equal to your own.

Several possibilities suggest themselves immediately. A contract with the company from which you bought your computer system is the most obvious. Another, possibly better, solution is an association of other local businessmen who all use the same type of system. Two variations on this idea are possible. First is a simple agreement to let any of the members use your system in the event of some major catastrophe. Of course, you will have similar access to their systems. The other variation is for the group as a whole to purchase a larger system that is compatible with all the members' systems. This will require much more planning and effort, but may provide a better long-term solution.

A local user's group like this provides you not only with site backup but also provides a forum in which to discuss problems the members may encounter in using their systems. You will also be able to provide a number of useful community services. For example, the group might want to hold adult or juvenile (eight-year-olds take to computers very well) classes on programming, or you might maintain mailing lists or accounting data for area churches and other charitable institutions.

**Making frequent backups will help protect your business from financial disaster due to human error or some minor machine malfunction.**

### Back Yourself Up

So far we've protected ourselves against the loss of vital data and against catastrophic system loss. What happens if *you* are lost? What if you're in an auto accident and get a broken leg and some cracked ribs? It's pretty hard to key in data while you're in a body cast. Your employees can probably run the business from day to day, taking orders or whatever, but can one or more of them run the computer? Train somebody — preferably several somebodies — and keep *complete* documentation. Sure, you probably have the documentation that came with the system when you bought it. You may also have the documentation that came with your software, but does all that paper really describe the day-to-day operation of your system? Does it tell where you keep your tapes? Does it tell when and why each program is run? Chances are it doesn't. It's easy to write this documentation. When you sit down to do the daily processing, have someone completely unfamiliar with the system take notes. Explain everything you do from getting the input documents together to turning the machine on to doing the backups to going home for the night.

You say you're never sick and don't drive a car? Reread my comment about people who don't do backups. About a week after writing the documentation (so the person you taught will forget what he saw you do), sit back and let the guy run the system himself. Whenever anything isn't quite right, correct the notes, and keep going through this cycle until anyone can run the system correctly from the documentation. Then type it up and make several copies. File some of the copies in easy-to-find places — like on top of the computer or in your top desk drawer. Put some other copies in safe places — like in the safe deposit box with the programs.

You should now be well prepared to plan your own backup scheme. A few hours spent in devising a secure system will certainly save many hours or days of frustration — particularly if you read this article and then forget about it.

Remember: There are three separate parts of the total system to be backed up. First — the software, i.e., your programs and data. Always use the best quality tapes available. They only cost a couple of dollars more than the garden-variety audio tapes and are more than worth it. Second — back up your hardware. Be certain that you will be able to run all your programs *somewhere* if your machine should crap out at the end of the month (they always blow up at the worst possible time). Third — back yourself up. No professional installation is ever so dependent on a single person that it cannot function without him. This final point is the most neglected of the three. In a way it is the most important. If you lose your inventory master you can always recover it the hard way. If your computer breaks down, you can always wait for repairs and then enter a lot of data all at once. If you're down, no one can replace you without proper documentation. You are the head of your business. That documentation will keep the body alive until you are able to resume normal functions.

For every computer system that fails, two or three sales will be lost. Failures make big news. Don't be famous. ■

# Small Business Software

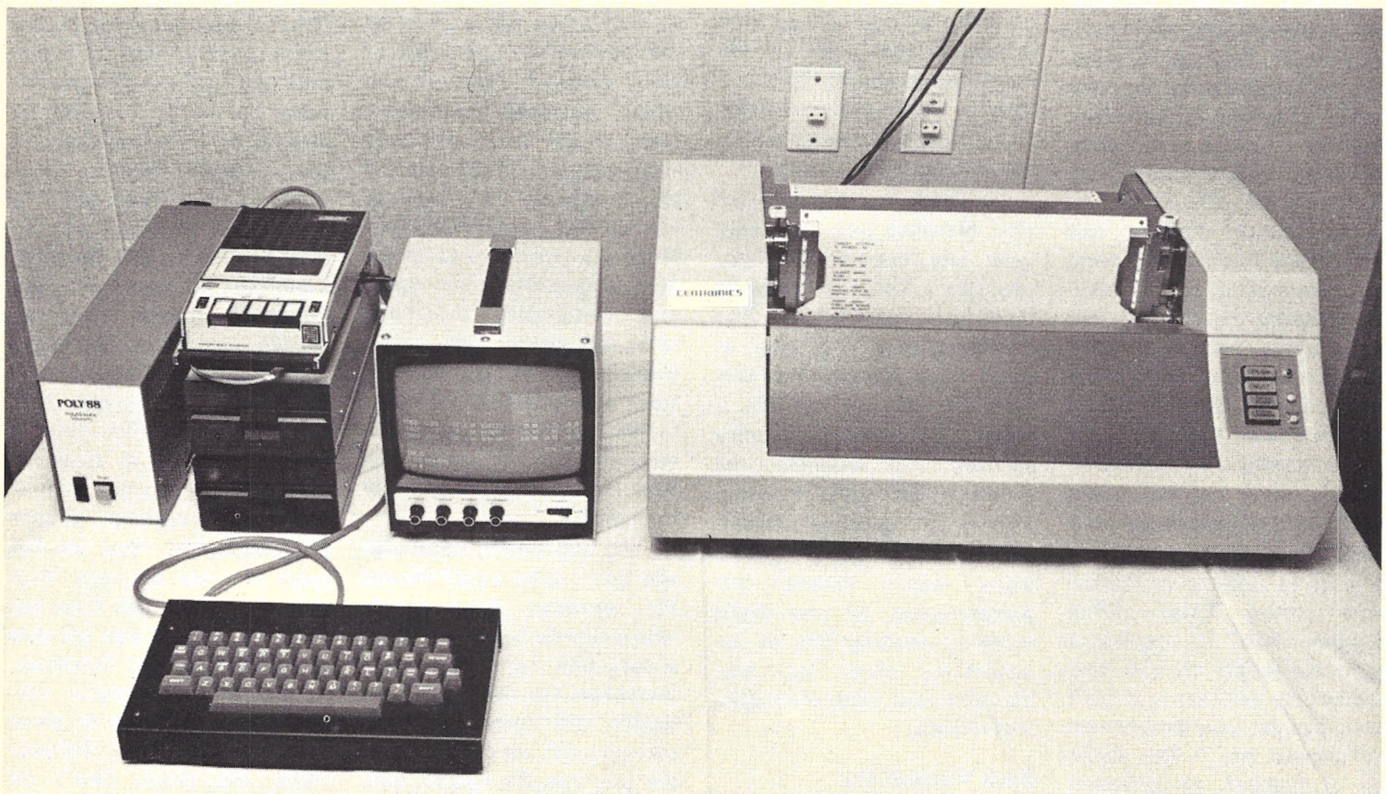
## Part 2: file-maintenance and active transaction programs

*Laurence A. McCaig*
*Microtec Computers Inc.*
*112 Elm Street*
*Newport ME 04953*

Last month I showed you the programs that are used to initially create the customer master and active accounts file. This month I will cover the file-maintenance programs, the sort program for the active transaction file and two new programs that were added to improve the simplicity of operation of the system.

Since I completed Part 1, North Star has come out with an improved version of their BASIC. This new version has one very useful feature that allows the internal linkage of programs. This increases the ease of operation so that under usual conditions only one program per system will actually have to be loaded by the operator. I will show you how this is accomplished, but first I want to make you aware of one very important thing: It is now mandatory that you have 24K of memory.

The new BASIC uses about 500 additional bytes of memory, and this doesn't leave much user area available if only 16K is being used. However, this is not much of a problem since the price of memory has become so low. George Morrow of Thinker Toys is offering 8K memory boards fully assembled and tested for $149; from what I have seen, they look like an excellent buy. I expect to see more good items coming from



his firm in the future.

**The Programs**

Take a look at Program 1 (WAKEUP). This little program is the entry point to the entire receivable system. Initially, this program asks, "Has my operator updated yesterday's files?" by looking at item P of record zero on the active transaction file. If the value of this item is zero, all is well, and the computer shoots off to execute Program 2 (ACREC047) (more on this in a minute). If item P contains the value 9999, there are still items on the transaction file that have not been updated, in which case the operator is given a message (see Fig. 1) and must choose to do one of two things.

It is possible that the operator interrupted the program flow intentionally in order to run the payroll or some other system. In this case, end-of-day processing is not to be done; after the operator responds "no," the computer will go on to Program 4 (START), which will be discussed later. If the operator decides that yesterday's processing has, in fact, not been done, he will answer "yes," and the following will occur.

First, Program 8 (ACREC055), which sorts the transaction file, is loaded; then the sales register and cash-receipts register will be printed. After all balances have been approved, the active file will be updated, after which the operator should again start at Program 1—as if it were the beginning of the day.

Now, if item P of record zero on the transaction file had a value of zero (the normal condition at the beginning of daily processing), Program 2 (ACREC047) will be loaded and executed. This program initializes the active transaction file; and that is its only purpose.

Next, Program 3 (ACREC040) is loaded and executed. This program, which is also part of the daily start-up procedure, updates the micro data base contained in the customer master and active file. Lines 90 through 120 zero all the pointers on the customer master (item A of this file). The active file is next read through sequentially, and the pointers in both files are updated. Item M of the active file is the data pointer to the customer master file. When this is completed, Program 4 (START) is loaded, and we are ready to take commands from the operator.

START is the second new program that has been added since Part 1. This program is used as the control point for all other programs. The operator is given several choices to make (see Fig. 2) that will determine which programs are to be run. All he need do is type the function code and hit RETURN. When the procedure that he selected has been run, control is again returned to the START program. This is one reason why I think the chaining feature is so important. It will make everything run so much smoother.

Now take a look at Program 5 (ACREC020). In order to perform maintenance on the customer master file, a work disk will be necessary. This disk must contain two data files; the first should be named WORK1 and contain 200 blocks; the second must be named WORK2 and contain 134 blocks. Also, set up on the accounts-receivable disk 1 a data file named CUSTRAN and containing 20 blocks. This is the file that will contain all additions and deletions. The program now allows changes to existing records, deletion of existing records or addition of new records. Data may be entered randomly, which makes it easier for the operator. However, the following limitations must be noted.

First, the combined total of additions and deletions per run should not exceed 25. If it does the message ENOUGH will be displayed, and the program will branch off into the sort routine. Although I have found that 25 entries are sufficient, it is possible to increase this limit. Suppose, for example, you want to increase the capacity

```
END OF DAY PROCESSING HAS NOT BEEN DONE FOR LAST DAY!
TYPE YES IF THIS IS TO BE DONE NOW.  IF YOU TYPE NO
ALL DATA ON THE TRANSACTION FILE WILL BE DELETED

     ***** PLEASE BE AWARE OF THIS ***** !!!!!

TYPE YOUR ANSWER!
```

*Fig. 1.*

```
MICROTEC ACCOUNTS RECEIVABLE SYSTEM
CODE    FUNCTION
  1 -   CUSTOMER FILE CREATION (INITIAL DATA ENTRY)
  2 -   CUSTOMER FILE MAINTENANCE (ADD DEL OR CHG)
  3 -   LIST CUSTOMER MASTER FILE
  4 -   ENTER DAILY ACCOUNT INFO (CHARGES, PAYMENTS,
        DEBIT MEMOS, AND CREDIT MEMOS) & SHOW ACCT BAL.
  5 -   END OF DAY PROCESSING (CASH RECIEPTS REPORT AND
        SALES REGISTER)                    [sic]
  6 -   ADD TODAYS ACCOUNTS TO THE ACTIVE ACCOUNT FILE
        (USE ONLY AFTER BALANCES HAVE BEEN CHECKED AND
        APPROVED)
  7 -   RUN INTEREST CALCULATION, AGED TRIAL BALANCE AND
        STATEMENTS
 99 -   END OF PROCESSING
  WHAT FUNCTION WOULD YOU LIKE ?0
```

*Fig. 2.*

```
                         MICROTEC COMPUTERS INC.
                         112 ELM STREET
                         NEWPORT, ME 04953          TEL 368-4434

TERMS:  NET 10, 1.5% OVER 30      MO PMT 50.00

 DATE    INV #   PART#    QTY DESCRIPTION     AMOUNT SLS TX    TOTAL

770531      0       0     .00 PREV BAL       716.06     .00   716.06
770629      0       0    5.00 KERO             2.50     .00   718.56
770705      0       0     .00 PAYMENT       -250.00     .00   468.56

                              TOTAL                           468.56
```

*Fig. 3.*

to 50 entries. Change the DIMX(25,1) statement in line 70 to DIM X(50,1), and change IF X1 = 25 in line 280 to IF X1 = 50. Furthermore, when a record is changed, the complete data for the customer must be entered—not just the item being changed. I should note here that there is no limitation on the number of changes that may be made in any one run.

OK, now I'll tell you a little bit about how the thing works. Changes are made directly to the customer master file, so there is no entry made on the transaction file. Therefore, if a flag at the end of the program that is turned on only if additions or deletions have been made can be checked, then the program that updates the customer master file may not have to be run. There is such a flag, R5, which is set in line 150 and then tested at the end of the data-entry portion in line 320.

When the customer number is entered, the customer master file is checked for a record on file that matches. If so, Q9 is set to 0; otherwise it is set to 1. This flag is used to make sure you are not trying to change or delete a record not on file, and also to make sure you are not trying to add a customer number already on file.

If either of these conditions exists, an error message will be displayed and the transaction will be ignored. Enough information has been presented to the program at this point so that if a delete is being performed, the record can be written on the transaction file and stored away in the file pointer table.

Lines 210 through 260 will be performed for additions and changes. When the operator has finished entering his data, he types an E at the transaction code entry (line 90), and the program will jump to line 320. Here, R5 is tested. If equal to zero, program START is reloaded; otherwise, table X is sorted, and the transactions are rewritten onto the back of the file in sequential order. Next, Program 6 (ACREC025) is loaded and executed.

Program 6 requires the use of the work disk previously mentioned. In line 60, the operator is requested to put this disk into drive two and then type RETURN. After this has been done, the program copies the current customer master file onto the work disk. Then this work file is written back onto the original file while the deleted items are dropped and the additions are added. When that's finished, the operator is requested to remove the work disk and replace it with the accounts-receivable disk 2. He is then asked if any more maintenance has to be performed. If his answer is yes, ACREC020 is again loaded and the process is repeated. Otherwise, program ACREC040 is loaded and the file pointers are updated.

Maintenance of the active accounts file is performed in a different manner; program 7 (ACREC050) accomplishes this. It is designed to be the program on-line in the computer most of the day. As a customer enters the door, his customer number is entered and a complete account history is displayed on the screen (see Fig. 3). Payments, charges, credits or debit memos may be applied directly to his account. This data is stored on the accounts transaction file until end-of-day processing, at which time the sales register and cash-receipts register are printed. The operator may interrupt this program at any time by typing 9999 for the customer number, and then reenter the program later in the day without any loss of data.

In Part 1, I said I would show you where to make modifications if you wanted to include several term codes. Line 190 is set up for term code A, and you can insert as many others as you use after this statement. The valid transaction codes are P for payment, C for charges, M for credit memos and D for debit memos. The statements are set so that if a P is entered in the description field, it will automatically be set to PAYMENT. Also, sales tax will be automatically calculated if a Y is entered when the program asks if this is to be done. Line 120 should be changed so that T5 contains the percentage of tax charged in your state. When you have completed entry of data for a particular customer, enter 9999 in the invoice number field, and the system will wait for another customer to be entered.

Programs 8 and 9 combine to form the sort program for the accounts transaction file. Program 8 reads the keys and file position of the active transaction file into a table of 100 items, sorts this table and then writes the sorted table out onto a work disk. This process continues until all data has been read in and sorted. The end result is a file containing blocks of data (keys and file pointers) sorted within themselves but not with each other. The next program (ACREC056) merges these blocks of data, two at a time until the final file contains one completely sorted block. This file is then read sequentially, and the transaction file is rewritten in the order provided by the file pointers on the work file. When the sort is completed, program ACREC060 is loaded and executed. This program produces the daily sales report and will be presented next time.

Also next time I will present the print routine, which is the only program that is written in assembly language. It is provided with an interface to BASIC in order to make it easy to use. I have included many useful features such as vertical tabbing, elongated character control, form feed, etc., and with the instructions that I'll include, you should be able to use it with other versions of BASIC. I'll also include the customer master list program, the cash-receipts program, sales program and merge of active transaction file to active file program. What I've presented here should keep you busy until then. ■

```
10 REM *--WAKEUP--
20 REM *--INITIAL LOAD PROGRAM FOR DAY--
30 REM *--COPYRIGHT AUG 25, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT ME 04953
50 REM
60 DIM F$(10)\DIM G$(1)\DIM H$(1)\K=0
70 OPEN#0, "ACTRAN, 2"\READ#0%63*K, L, M, N, O, P, Q, F$, G$, R, S, I, H$
75 CLOSE #0
80 IF P<>9999 THEN 90 ELSE 160
90 !CHR$(12)
100 !"END OF DAY PROCESSING HAS NOT BEEN DONE FOR LAST DAY!"
110 !"TYPE YES IF THIS IS TO BE DONE NOW.  IF YOU TYPE NO"
120!"ALL DATA ON THE TRANSACTION FILE WILL BE DELETED"
125 !\!"      ***** PLEASE BE AWARE OF THIS ***** !!!!!"\!
130 INPUT "TYPE YOUR ANSWER!", Z$
140 IF Z$="NO" THEN CHAIN "START"
150 IF Z$="YES"THEN CHAIN "ACREC055"
155 !CHR$(12), "YOU DIDN'T ANSWER YES OR NO!"\!\!\GOTO 100
160 CHAIN "ACREC047"
170 END
READY
```

*Program 1.*

*Program 2.*

```
10 REM *--ACRECO47--
20 REM *--SET UP ACTIVE TRANSACTION FILE
30 REM *--COPYRIGHT JULY 28, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT MAINE 04953
50 DIM F1$(10)\DIM G1$(1)\DIM H1$(1)\I1=0
60 OPEN #0,"ACTRAN,2"
70 O1=0\K1=0
80 WRITE #0%63*K1,L1,M1,N1,O1,P1,Q1,F1$,G1$,R1,S1,I1,H1$
90 CLOSE #0
100 CHAIN "ACRECO40"
READY
```

*Program 3.*

```
10 REM *--ACRECO40-- UPDATES FILE POINTERS ON THE CUST
20 REM *--MASTER AND ACTIVE ACCOUNT FILE.
30 REM *--COPYRIGHT JULY 6,1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953
50 !CHR$(12)\!TAB(5),"I'VE GOT TO ARRANGE MY POINTERS"
60 !"IT WILL ONLY TAKE A FEW MINUTES"
70 DIM B$(15)\DIM C$(20)\DIM D$(20)\DIM F$(10)
80 DIM G$(1)\DIM H$(1)\DIM A$(8)\DIM E$(8)\I=0
90 OPEN #0,"CUSTMST"\OPEN#1,"ACTIVE,2"
100J=0\K=0
110GOSUB540\IFJ=0THENJ1=G\A=0\GOSUB560\IFJ=J1THEN130\J=J+1
120 GOTO 110
130 K=0\L1=0
140!CHR$(12)\!TAB(5),"HALF WAY THERE"
150 L1=0
160 GOSUB 320
170 K1=0
180 K=K+1
190 GOSUB 320
200 M=0\A9=L
210 IF L=L1 THEN 270
220 A3=K
230 GOSUB 360
240 A=A3
250 IF Q9<>1 THEN GOSUB 560
260 IF Q9=0 THEN M=J
270 GOSUB 340
280 L1=L
290 IF K=K1 THEN 580
300 GOTO 180
310 A9=L
320 READ #1%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$
330 RETURN
340 WRITE #1%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$,NOENDMARK
350 RETURN
360 J=0\Q9=0
370 GOSUB 540
380 H9=G\L9=0
390 R9=H9
400 GOTO 450
410 R9=(H9-L9)/2+L9
420 R9=INT(R9)
430 IF H9=R9 THEN 520
440 IF L9=R9 THEN 520
450 J=R9\GOSUB 540
460 IF A9=F THEN 530
470 IF F<A9 THEN 500
480 H9=R9
490 GOTO 410
500 L9=R9
510 GOTO 410
520 Q9=1
530 RETURN
540 READ #0%96*J,A,F,A$,B$,C$,D$,E$,G
550 RETURN
560WRITE #0%96*J,A,F,A$,B$,C$,D$,E$,G,NOENDMARK
570 RETURN
580 CLOSE #0
590 CLOSE #1
600!"ALL DONE"
610 CHAIN "START"
READY
```

*Program 4.*

```
10 REM *--START--
20 REM *--CONTROL PROGRAM FOR ACCOUNTS RECEIVABLE--
30 !
40 !TAB(2) ,"MICROTEC ACCOUNTS RECEIVABLE SYSTEM"
50 !TAB(2),"CODE   FUNCTION"
60 !TAB(4),"1 -  CUSTOMER FILE CREATION (INITIAL DATA ENTRY)"
70 !TAB(4),"2 -  CUSTOMER FILE MAINTENANCE CHG)"
80 !"     ","3 -  LIST CUSTOMER MASTER FILE           "
90 !CHR$(27),"     ","4 -  ENTER DAILY ACCOUNT INFO (CHARGES, PAYMENTS,"
100 !TAB(9) ,"DEBIT MEMOS, AND CREDIT MEMOS) & SHOW ACCT BAL."
110 !TAB(4),"5 -  END OF DAY PROCESSING",
120!" (CASH RECIEPTS REPORT AND"\!TAB(9) ,"SALES REGISTER)"  [sic]
130!TAB(4),"6 -  ADD TODAYS ACCOUNTS TO THE ACTIVE ACCOUNT ",
140!"FILE"\!TAB(9) ,"(USE ONLY AFTER BALANCES HAVE BEEN ",
150!"CHECKED AND"\!TAB(9) ,"APPROVED)"
160!TAB(4),"7 -  RUN INTEREST CALCULATION, AGED TRIAL BALANCE",
170 !" AND"\!TAB(9) ,"STATEMENTS"
180!TAB(3),"99 -  END OF PROCESSING"
190 !TAB(5),"WHAT FUNCTION WOULD YOU LIKE ",\INPUT T
200 !
210 IF T=1 THEN CHAIN "ACRECO10"
220 IF T=2 THEN CHAIN "ACRECO20"
230 IF T=3 THEN CHAIN "ACRECO30"
```

```
240 IF T=4 THEN CHAIN "ACRECO50"
250 IF T=5 THEN CHAIN "ACRECO55"
260 IF T=6 THEN CHAIN "ACRECO80"
270 IF T=7 THEN CHAIN "ACRECO85"
280 !CHR$(12)\!"THANK YOU.  ACCOUNTS RECEIVABLES DONE."
290 END
READY
```

*Program 5.*

```
10 REM *--ACRECO20--
20 REM *--CUSTOMER MASTER MAINTENANCE PROGRAM PART 1--
30 REM *--COPYRIGHT JUL 28, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953--
50 REM
60GOSUB520\GOSUB530
70DIMB$(15)\DIMC$(20)\DIMD$(20)\DIMX(25,1)\DIMT$(1)\DIMA1$(8)
80DIMB1$(15)\DIMC1$(20)\DIMD1$(20)\DIME1$(8)\GOSUB480
90INPUT"TRAN CD A=ADD C=CHG D=DEL E=END ",T$(1,1)
100IFT$="E"THEN320
110IFT$="Z"THEN90\IFT$<"A"THEN130\IFT$>"D"THEN130
120IFT$="B"THEN130\GOTO140
130!"INVALID CODE"\GOTO090
140INPUT"CUST# ",A9\GOSUB610
150IFT$="C"THEN160\R5=1
160IFT$="A"THEN180\IFT$="C"THEN170\IFT$="D"THEN170
170IFQ9<>1THEN190\!"NOT FND"\GOTO090
180IFQ9=1THEN190\!"DUPLCT"\GOTO090
190F1=A9\IFQ9=1THEN200ELSEGOSUB580
200IFT$="D"THEN270
210INPUT"1ST NM ",A1$(1,8)\IFA1$(1,2)="ZZ"THEN90\F1=A9
220INPUT"LST NM ",B1$(1,15)\IFB1$(1,2)="ZZ"THEN90
230INPUT"1ST ADR LN ",C1$(1,20)\IFC1$(1,2)="ZZ"THEN90
240INPUT"2ND ADR LN ",D1$(1,20)\IFD1$(1,2)="ZZ"THEN90
250INPUT"TEL # ",E1$(1,8)\IFE1$(1,2)="ZZ"THEN90
260INPUT"MO PMT AMT ",G1\IFG1=9999THEN90\IFT$="C"THEN300
270J1=J1+1\X1=X1+1\X(X1,0)=A9\X(X1,1)=J1\GOSUB480
280IFX1=25THEN290\GOSUB560\GOTO090
290!"ENOUGH"\GOTO320
300F=F1\A=0\A$=A1$\B$=B1$\C$=C1$\D$=D1$\E$=E1$
310G=G1\GOSUB500\GOSUB560\GOTO090
320IFR5=0THEN470\X(0,0)=X1\G1=J1+1\J1=0\GOSUB480\GOSUB540
330 GOSUB550
340X9=X1\X1=1\X8=X9\X2=1
350IFX(X1,0)>X(X8,0)THEN380\IFX(X1,0)<X(X2,0)THEN400
360X1=X1+1\IFX1<>X8THEN350\X2=X2+1\X8=X8-1\IFX2>=X8THEN420
370X1=X2\GOTO0350
380Y=X(X8,0)\Y1=X(X8,1)\X(X8,0)=X(X1,0)\X(X8,1)=X(X1,1)
390X(X1,0)=Y\X(X1,1)=Y1\GOTO0360
400Y=X(X2,0)\Y1=X(X2,1)\X(X2,0)=X(X1,0)\X(X2,1)=X(X1,1)
410X(X1,0)=Y\X(X1,1)=Y1\GOTO0360
420GOSUB530\J1=0\GOSUB490\J5=G1+1\GOSUB510\J3=G1\X1=1
430J1=X(X1,1)\GOSUB490\J5=J5+1\GOSUB510\IFX1=X9THEN450
440X1=X1+1\GOTO0430
450G1=J5\J5=J3\GOSUB510\GOSUB550
460 CHAIN "ACRECO25"
470 CHAIN "START"
480WRITE#1%94*J1,T$,F1,A1$,B1$,C1$,D1$,E1$,G1,NOENDMARK\RETURN
490READ #1%94*J1,T$,F1,A1$,B1$,C1$,D1$,E1$,G1\RETURN
500WRITE#0%96*J,A,F,A$,B$,C$,D$,E$,G,NOENDMARK\RETURN
510WRITE#1%94*J5,T$,F1,A1$,B1$,C1$,D1$,E1$,G1,NOENDMARK\RETURN
520OPEN#0,"CUSTMST"\RETURN
530OPEN#1,"CUSTRAN"\RETURN
540CLOSE#0\RETURN
550CLOSE#1\RETURN
560C1$="                    "\D1$=C1$\B1$=C1$\A1$=C1$\E1$=C1$
570 RETURN
580!"     ",%4I,F,"     ",A$," ",B$\!TAB(12),C$
590!TAB(12),D$," TEL ",E$
600!"MO PMT ",G\RETURN
610Q9=0\J=0\GOSUB690\H9=G\L9=0
620R9=H9\GOTO0640
630R9=(H9-L9)/2+L9\R9=INT(R9)\IFH9=R9THEN670\IFL9=R9THEN670
640J=R9\GOSUB690\IFA9=FTHEN680\IFF<A9THEN660\H9=R9
650GOTO0630
660L9=R9\GOTO0630
670Q9=1
680RETURN
690READ#0%96*J,A,F,A$,B$,C$,D$,E$,G\RETURN
READY
```

*Program 6.*

```
10 REM *--ACRECO25--
20 REM *--UPDATE CUSTOMER MASTER--
30 REM *--COPYRIGHT JULY 15, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953
50 !CHR$(12)
60 INPUT"PUT WORK DISK IN DRIVE 2 THEN PRESS RETURN",Z$
70!CHR$(12)
80DIMA$(8)\DIMB$(15)\DIMC$(20)\DIMD$(20)\DIME$(8)\DIMT$(1)
90 DIM B1$(15)\DIM C1$(20)\DIM A1$(8)\DIM E1$(8)
100GOSUB340\GOSUB350
110GOSUB280\IFJ=0THENJ9=G\GOSUB310\IFJ=J9THEN120\J=J+1\GOTO0110
120CLOSE#1\GOSUB350\GOSUB360\GOSUB320\GOSUB330\J2=G1\GOSUB330
130J8=G1\J9=G\CLOSE #0\GOSUB 340
140J=0\GOSUB290\GOSUB370\GOSUB400
150IFF+F1=19998THEN210
160IFF<F1THEN170\GOTO0180
170J=J+1\GOSUB290\GOSUB400\GOTO0150
180IFF1=FTHEN190\IFT$="D"THEN200\J=J+1\GOSUB300\GOTO0200
190IFT$<>"D"THEN200\GOSUB400
200GOSUB370\GOTO0150
210G=J\J=0\GOSUB290\CLOSE#0\CLOSE#1\CLOSE#2
```

```
220PRINT"DONE.  PUT ACREC2 BACK IN DRIVE 2.",
230INPUT1" THEN PRESS RETURN",Z$
240INPUT"DO YOU HAVE MORE MAINTENANCE? YES OR NO ",Z$
250IFZ$="YES" THEN CHAIN "ACRECO20"
260IFZ$="NO" THEN CHAIN "ACRECO40"
270GOTO 240
280READ#0%96*J,A,F,A$,B$,C$,D$,E$,G\RETURN
290WRITE#0%96*J,A,F,A$,B$,C$,D$,E$,G,NOENDMARK\RETURN
300WRITE#0%96*J,A1,F1,A1$,B1$,C1$,D1$,E1$,G1\RETURN
310WRITE#1%96*J,A,F,A$,B$,C$,D$,E$,G,NOENDMARK\RETURN
320READ#1%96*J1,A,F,A$,B$,C$,D$,E$,G\RETURN
330READ#2%94*J2,T$,F1,A1$,B1$,C1$,D1$,E1$,G1\RETURN
340OPEN#0,"CUSTMST"\RETURN
350OPEN#1,"WORK1,2"\RETURN
360OPEN#2,"CUSTRAN"
370IFF1=9999THEN390\J2=J2+1\IFJ2<=J8THEN380\F1=9999\GOTO390
380GOSUB330
390RETURN
400IFF=9999THEN420\J1=J1+1\IFJ1<=J9THEN410\F=9999\GOTO420
410GOSUB320
420RETURN
READY
```

---

*Program 7.*

```
10 REM *--ACRECO50--
20 REM *--DATA ENTRY PROGRAM FOR DAILY ACCOUNT ACTIVITY--
30 REM *--COPYRIGHT JUL 28, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953--
50 OPEN#0,"CUSTMST"
60 OPEN#1,"ACTIVE,2"
70 OPEN#3,"ACTRAN,2"
80 DIMB$(15)\DIMC$(20)\DIMD$(20)\DIMF$(10)\DIMG$(1)
90 DIMH$(1)\DIMG1$(1)\DIMH1$(1)\DIMF1$(10)\DIMX$(20)
100K1=0\GOSUB820\K1=01\GOSUB820\DIMR$(1)\I=0
110K=0\GOSUB680\06=0
120J=0\K=0\T5=.05
130INPUT"CUST# ",A9\Z1=0\IFA9=9999THEN830\T=0\GOSUB720
140IFQ9<>1THEN170
150!"NOT ON FILE"
160GOTO130
170K=A\GOSUB680
180L5=L
190IFG$="A"THENX$="NET 10, 1.5% OVER 30"
200!TAB(25),A$," ",B$
210!TAB(25),C$
220!TAB(25),D$,"      TEL ",E$
230!"TERMS: ",X$,"      MO PMT ",%5F2,G
240IFA<>0THEN270
250!"THIS CUSTOMERS BALANCE IS $.00"
260GOTO420
270!" DATE   INV #  PART#    QTY DESCRIPTION   AMOUNT",
280!"SLS TX   TOTAL"
290IFH$="M"THEN320
300IFH$="P"THEN320
310GOTO330
320S=0-S\R=0-R
330T=R+S+T
340!%6I,N,"  ",%5I,O,"  ",%5I,P,"  ",%6F2,Q,"  ",F$,"    ",
350!%7F2,R,"  ",%6F2,S,"  ",%7F2,T
360Z1=1
370K=K+1\IFK=K5THEN410
380IFK>O6THEN410
390GOSUB680
400IFL<>L5THEN410\GOTO290
410!\!TAB(50),"TOTAL ",%7F2,T
420IFZ1=0THENINPUT"TERM CD ",G1$(1,1)
430!"TO END INV#=9999"
440INPUT"INV# ",O1
450GOSUB670
460IFO1=9999THEN120
470INPUT"PART# ",P1
480IFP1=9999THEN440
490INPUT"QTY ",Q1
500IFQ1=9999THEN440
510INPUT"DESC ",F1$(1,10)\IFF1$(1,3)="P  "THEN520ELSE530
520F1$(1,10)="PAYMENT"\H1$(1,1)="P"
530IFF1$(1,2)="ZZ"THEN440
540INPUT"AMT ",R1
550IFR1=9999THEN440
560S1=0\IFH1$="P"THEN630
570INPUT"Y FOR SLS TAX ",R$\IFR$="Z"THEN440\IFR$="Y"THEN590
580GOTO600
590S1=R1*T5\!"SLS TX IS ",%7F2,S1
600INPUT"TRAN CD ",H1$(1,1)\IFH1$="Z"THEN440
610IFH1$="C"THEN630\IFH1$="D"THEN630\IFH1$="M"THEN630
620IFH1$="P"THEN630\!"INVALID TRAN CD "\GOTO600
630K1=K1+1\GOSUB700
640M1=J
650GOSUB670
660K2=K1\K1=0\O1=K2\GOSUB700\K1=K2\GOTO430
670F1$(1,10)="          "\H1$(1,1)=" "
680READ#1%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$
690RETURN
700WRITE#3%63*K1,L1,M1,N1,O1,P1,Q1,F1$,G1$,R1,S1,I,H1$,NOENDMARK
710RETURN
720J=0\Q9=0\GOSUB810\H9=G\L9=0\R9=H9\GOTO750
730R9=INT((H9-L9)/2+L9)
740IFH9=R9THEN790\IFL9=R9THEN790
750J=R9\GOSUB810
760IFA9=FTHEN800
770IFF<A9THEN780\H9=R9\GOTO730
780L9=R9\GOTO730
790Q9=1
800RETURN
810READ#0%96*J,A,F,A$,B$,C$,D$,E$,G\RETURN
820READ#3%63*K1,L1,M1,N1,O1,P1,Q1,F1$,G1$,R1,S1,I,H1$\RETURN
830CLOSE#0\CLOSE#1\CLOSE#3\CHAIN "START"
840END
READY
```

*Program 8.*

```
10 REM *--ACREC055--
20 REM *--PHASE 1 SORT FOR ACTRAN--
30 REM *--COPYRIGHT JULY 15, 1977, MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953
50 REM
60 INPUT "PUT WORK DISK IN DRIVE 2, WHEN READY TYPE 'GO' ",C$
70 IF C$="GO" THEN 80 ELSE !"INVALID COMMAND-RETYPE"\GOTO60
80 GOSUB 370\GOSUB 380\W=1\K=0
90 DIM F$(10)\DIM G$(1)\DIM H$(1)\DIM X(100,1)
100 Z1=0\GOSUB 240\GOSUB 230\K1=0\K3=0\Z1=1
110 K2=K3+1\K3=K2+99\X1=1
120 FOR K=K2 TO K3\GOSUB 230\X(X1,0)=L\X(X1,1)=K
130 IF K<K1 THEN 140\Z=1\EXIT 150
140 X1=X1+1\NEXT
150 REM
160 X(0,0)=L\X(0,1)=X1\GOSUB 250\X9=X9+1
170 FOR J=0 TO J5\Z1=Z1+1
180 Z2=X(J,0)\Z3=X(J,1)\GOSUB 240\NEXT\Z2=X9\Z3=100
190 Y9=Z1\Z1=1\GOSUB 240\Z1=Y9\IF Z=1 THEN 200\GOTO 110
200 Z2=Z1\Z1=0\Z3=0\GOSUB 240
210 GOSUB 390\GOSUB 400
220 CHAIN "ACREC056"
230 READ #0%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$\RETURN
240 WRITE #1%11*Z1,Z2,Z3,NOENDMARK\RETURN
250 REM --SORT RTN--
260 X2=1\X3=X1\X4=X2\IF X3=101 THEN X3=100
270 X4=X2
280 IF X(X4,0)>X(X3,0) THEN 310\IF X(X4,0)<X(X2,0) THEN 330
290 X4=X4+1\IF X4>X3 THEN 300\GOTO 280
300 X2=X2+1\X3=X3-1\IF X2>=X3 THEN 350\GOTO 270
310 Y9=X(X3,0)\X(X3,0)=X(X4,0)\X(X4,0)=Y9
320 Y9=X(X3,1)\X(X3,1)=X(X4,1)\X(X4,1)=Y9\GOTO 290
330 Y9=X(X2,0)\X(X2,0)=X(X4,0)\X(X4,0)=Y9
340 Y9=X(X2,1)\X(X2,1)=X(X4,1)\X(X4,1)=Y9\GOTO 290
350 J5=X1\IF X1=101 THEN J5=100
360 X(0,0)=X(J5,0)\X(0,1)=J5\RETURN
370 OPEN #0,"ACTRAN"\RETURN
380 OPEN #1,"WORK1,2"\RETURN
390 CLOSE #0\RETURN
400 CLOSE #1\RETURN
READY
```

*Program 9.*

```
10 REM *--ACREC056--
20 REM *--PHASE TWO SORT FOR ACCTS TRANSACTION FILE--
30 REM *--COPYRIGHT JULY 15, 1977 BY MICROTEC COMPUTERS INC.
40 REM *--112 ELM STREET, NEWPORT, MAINE 04953
50 REM *
60DIM F$(10)\DIM G$(1)\DIM H$(1)\GOSUB440\GOSUB450\W=1
70X1=0\GOSUB540\X1=1\GOSUB540\A1=0\B1=0\A2=0\Z1=1\GOSUB520
80B=Z2\Z1=2
90GOSUB520\X6=Z3\A1=Z1+Z3\IFB=1THEN 110\Y1=A1+1\GOSUB530
100X6=X6+Y3\A2=Y1+Y3\GOTO120
110Y2=9999\A4=1
120REM
130GOSUB380\X1=X1+1\X4=X1\GOSUB410
140IFZ2<=Y2THEN160\X2=Y2\X3=Y3\X1=X1+1\GOSUB540\GOSUB410
150GOTO170
160X2=Z2\X3=Z3\X1=X1+1\GOSUB540\GOSUB380
170A5=A3+A4\IFA5<>2THEN140
180B1=B1+1\X3=X6\GOSUB550
190IFB=1THENB=0ELSEB=B-2\A3=0\X6=0
200A4=0\Z1=Y1\IF B=0 THEN 210 ELSE 90
210IFW=1THENW=2ELSEW=1
220X1=0\IFY1>Z1THENX3=Y1-1ELSEX3=Z1-1\GOSUB540\X1=1\X2=B1
230GOSUB540\GOSUB480\GOSUB490\IFB1=1THEN260\IFW=1THEN250
240GOSUB460\GOSUB470\GOTO70
250GOSUB440\GOSUB450\GOTO70
260IFW=1THEN270\GOSUB460\GOSUB470\GOTO280
270GOSUB440\GOSUB450
280GOSUB500\Z1=2\GOSUB520\A1=Z1+Z3\K1=0\GOSUB580
290Z1=Z1+1\IFZ1>A1THEN310\GOSUB520\K=Z3\GOSUB560\K1=K1+1
300GOSUB580\GOTO290
310O=K1\K1=0\GOSUB580\GOSUB480\GOSUB490\K=1
320IFW=1THENGOSUB460\IFW=2THENGOSUB440\K1=0
330GOSUB590\IFK1=0THENK9=0\K=K1\GOSUB570\IFK1=K9THEN350
340K1=K1+1\GOTO330
350GOSUB480\GOSUB510
360INPUT"END SRT PUT ACREC DSK2 IN UNIT 2 THEN TYPE RETURN",B7$
370 CHAIN "ACREC060"
380IFA3=1THEN400\Z1=Z1+1\IFZ1>A1THEN390\GOSUB520\GOTO400
390A3=1\Z2=9999
400RETURN
410IFA4=1THEN430\Y1=Y1+1\IFY1>A2THEN420\GOSUB530\GOTO430
420A4=1\Y2=9999
430RETURN
440OPEN#0,"WORK1,2"\RETURN
450OPEN#1,"WORK2,2"\RETURN
460OPEN#0,"WORK2,2"\RETURN
470OPEN#1,"WORK1,2"\RETURN
480CLOSE#0\RETURN
490CLOSE#1\RETURN
500OPEN#2,"ACTRAN"\RETURN
510CLOSE#2\RETURN
520READ#0%11*Z1,Z2,Z3\RETURN
530READ#0%11*Y1,Y2,Y3\RETURN
540WRITE#1%11*X1,X2,X3,NOENDMARK\RETURN
550WRITE#1%11*X4,X2,X3,NOENDMARK\RETURN
560 READ#2%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$\RETURN
570WRITE#2%63*K,L,M,N,O,P,Q,F$,G$,R,S,I,H$,NOENDMARK\RETURN
580WRITE#1%63*K1,L,M,N,O,P,Q,F$,G$,R,S,I,H$,NOENDMARK\RETURN
590 READ#0%63*K1,L,M,N,O,P,Q,F$,G$,R,S,I,H$\RETURN
READY
```

*John Blankenship*
*DeVry Institute of Technology*
*828 W. Peachtree N.W.*
*Atlanta GA 30308*

# Expand Your KIM!

## *Part 4: a TTY substitute*

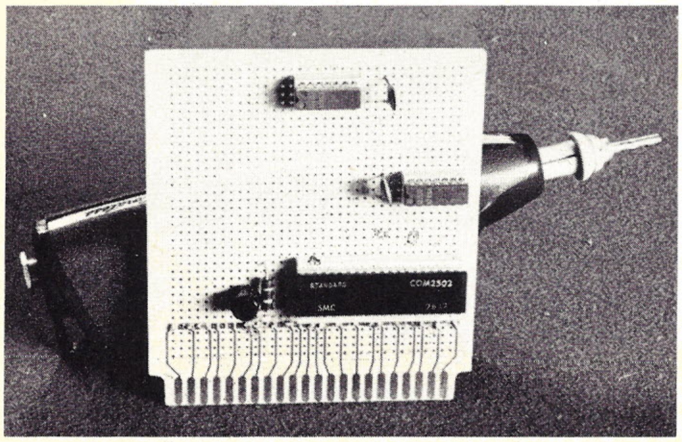


*Photo 1. This $10 circuit makes an SWTP keyboard and PR-40 printer act like a miniature Teletype.*

Even if you are not building the KIM-1 System described in the first three parts of this series, you may find this article interesting and useful.

The first section is devoted to converting the SWTP keyboard and PR-40 printer into a low-cost ($310) teletypewriter. The latter portion of the article describes a method for adding an external keyboard and display to KIM.

In the KIM-1 System, many useful subroutines, including I/O, are provided in ROM. These routines operate over the KIM 20 mA TTY interface lines. The cost of even a reconditioned TTY was prohibitive for me, and besides, the PR-40 printer was better suited to my requirement of portability.

To take full advantage of the ROM I/O routines, and to avoid using KIM's two nondedicated I/O ports, I set out to construct a 20 mA serial interface for the PR-40 and the SWTP keyboard.

### Overview

Since the interface would deal primarily with TTL signals on both ends, I eliminated the optical isolation often found on other interfaces. My baud-rate frequency would be far from critical, since KIM automatically adjusts its software timing to match the incoming signal. These considerations made possible a simple interface consisting of only three chips and one transistor.

The advantages of the KIM self-adjusting I/O routines are obvious, but, as it turns out, disadvantages are also present. The KIM-1 System utilizes a Dazzler for its video display (see last month's article), which puts the processor into a hold approximately 20 percent of the time so it may gain access to the bus for DMA. This delay creates the probability that the characters transferred during Dazzler operation will have faulty timing.

In addition, during observation of the above symptoms I found that the cable positions and mainframe wiring placement held potential noise problems from the Dazzler that could affect serial transfers. There are many methods of correcting either problem, but most corrections are more applicable to mass production where the designer will know the location of every wire and the length of each cable.

Not having this control over systems built by *Kilobaud* readers, I chose to shut off the Dazzler during all transfers of serial data, thus eliminating any possible timing or noise problems that might be introduced by the DMA operations.

Stopping the Dazzler naturally causes the video display to blank during data transfers; but since I'm transferring data at just over 5200 baud, the display only blanks for 2.1 ms, which is not noticeable.

This high data rate does provide a potential problem with the printer during carriage returns. The printer has a 40-character buffer memory and can accept data faster than any microprocessor can send it. However, during the carriage return (about one second), the inputs to the buffer are locked out, and any data transferred at that time is lost.

Most systems (including KIM) correct this problem by sending several nonfunctioning characters, such as nulls, after each carriage return, just to take up time. Although this works well for the standard TTY rate of 110 baud, it would take nearly 500 nulls for a one-second delay at 5200 baud.

I corrected this situation by putting the processor into a hold (thus stopping transfers) whenever a carriage return was in effect. Although I lose one second of processor time during carriage returns, I can load the buffer memory in only 85 ms as opposed to four seconds required at 110 baud. Obviously, I have increased my throughput time by as much as 500 percent with this method. One function of the interface is to convert the parallel data from the keyboard to compatible serial data for KIM. Compatible, in this case, means

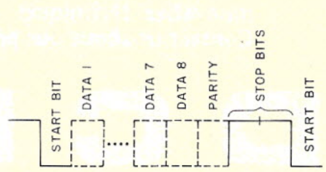



*Fig. 1. Timing detail for serial transfers.*

adding a start bit and two stop bits to each data word.

The serial data line normally remains at the high (logic 1) level, and a high-to-low transition (the start bit) is used to indicate the beginning of a new word of data. The high and low levels are often referred to, respectively, as marks and spaces.

Two stop bits (marks) follow the actual data. It should be noted that all words contain the same number of data bits, and the stop bits are *not* used to indicate the end of a word. The stop bits have two functions. First, if either of the stop bits happens to be a zero, a framing error (timing) is probably indicated. Second, if the data words are following each other without an elapsed time between them, the mark condition of the stop bits assures that the negative edge of the start bit will be properly recognized. If this is not clear, imagine two words without stop bits in which the last bit of word one was a space. Obviously, if a mark-to-space transition is to be recognized, the line must return to the mark condition before the second word. Fig. 1 shows the timing for the serial data.

The second function of the interface is exactly opposite of the parallel-to-serial conversion just described. It is to convert the serial data (coming from KIM) to the parallel data required by the printer, and the generation of a strobe to initiate the actual parallel transfer.

Generally, the above functions could be implemented with shift registers under proper control. Fortunately, the shift registers and the control circuitry are available in a single chip called a universal asynchronous receiver transmitter (UART).

I chose a 2502 UART because of its low cost, but most any will do if you study its data sheets and utilize it properly. Table 1 lists the pin usages for the 2502. Just remember that the UART is simply doing serial-to-parallel and parallel-to-serial conversions.

Fig. 2 is a page from my notebook showing the schematic of the interface, component placement and pinout designations. The circuit is built on a 44-pin board (see Photo 1).

### How It Works

The operation of the interface is simple: The 7492 is used as a divide-by-12 counter to lower the frequency applied to the UART.

The serial data coming from KIM is from an open collector NAND gate. When a logic 1 is being output to the printer, this gate has a low output, thus sinking the 20 mA signal provided by the printer return. I shorted the gate output and the return together (pins 6 and 8 on the interface card) causing the gate to act as standard TTL, except that the signal is inverted. I used gate A1 (see schematic) to invert this signal so the UART was receiving the correct information.

The UART converts the serial data from A1 to parallel and sends it to the printer under control of the high-level strobe (UART pin 19). Gate A2 converts this signal to the low-level strobe required by the printer.

The keyboard inputs its parallel data by pulsing the UART pin 23. The serial output is used to control the conduction of transistor $Q_1$, which can be almost any NPN transistor. The transistor simply makes and breaks the 20 mA input loop on KIM.

Gate A3 inverts the low-level KIM reset to a high-level pulse for the UART.

Gate A4 is used as a buffer (pin 2 of the interface card is shorted to ground when the Dazzler power switch is off). Since pin 2 connects to the S-100 clear, *either* the Dazzler power switch or the depression of a key will cause the Dazzler to halt the DMA operations, thus preventing possible noise and timing problems as described earlier. Naturally, since the Dazzler is turned off when a key is pressed, it must be turned back on when the character

| Pin | Function |
|---|---|
| 1 | 5 volt supply. |
| 2 | -12 volt supply. |
| 3 | Ground. |
| 4 | Grounded to keep parallel outputs enabled. |
| 5-12 | Parallel outputs. |
| 13 | High-level output indicates parity error (not used). |
| 14 | High-level output indicates framing error (not used). |
| 15 | High-level output indicates parallel outputs are not taken before new character is received (not used). |
| 16 | Grounded to enable outputs of pins 13, 14, 15, 19 and 22. |
| 17 | Clock input (output baud rate is 1/16 of this frequency). |
| 18 | Low-level input resets pin 19 to a low level. |
| 19 | High-level output indicates the parallel outputs are valid. |
| 20 | Serial input. |
| 21 | High-level pulse causes Master Reset. |
| 22 | High level indicates parallel input buffer empty ( not used). |
| 23 | Low-level pulse enters parallel data. |
| 24 | High-level output between serial transmissions (not used). |
| 25 | Serial output. |
| 26-33 | Parallel inputs. |
| 34 | High level enables pins 35, 36, 37, 38 and 39. |
| 35 | High level eliminates parity. |
| 36 | High level sets two stop bits. |
| 37-38 | High levels indicate eight-bit character. |
| 39 | Indicates even or odd parity (not used). |
| 40 | Clock input (must be 16 times baud rate). |

*Table 1. Pin functions for the 2502 UART.*

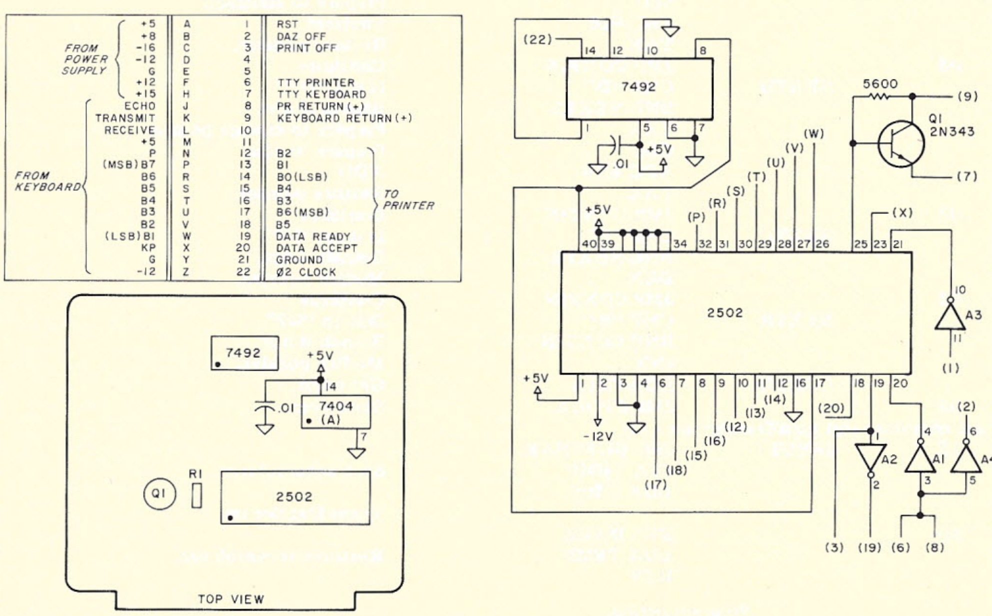



*Fig. 2. Interface schematic.*

has been transferred. The software example later in this article shows how this can be accomplished.

Note that when pin 3 of the interface card is shorted to ground by the Print power switch, transfers to the printer will be inhibited because the UART strobe signal is eliminated.

If you are constructing the KIM-1 System, do not pass lightly over these switch configurations since they provide useful functions that are not readily obvious.

Since the four TTY lines from KIM also connect to banana plugs on the backplane, an auxiliary TTY or video display may be easily connected to the system. The only requirement is that the

| Address | Instruction | | | Label | Mnemonic | Comment |
|---|---|---|---|---|---|---|
| 0300 | D8 | | | INIT | CLD | Select binary mode. |
| 01 | A9 | 00 | | | LDA #$00 | |
| 03 | 85 | 00 | | | STA Color | Clear color. |
| 05 | A9 | 10 | | | LDA #$10 | Sets Dazzler to a |
| 07 | 8D | 0F | 80 | | STA DAZ1 | 32 x 32 color mode. |
| 0A | A9 | 90 | | | LDA #$90 | Sets starting address |
| 0C | 8D | 0E | 80 | | STA DAZ2 | for display to page 20. |
| ; | Stores color in all of display area. | | | | | |
| 0F | A2 | FF | | ALL | LDX #$FF | Initialize Index X. |
| 11 | A5 | 00 | | ALL1 | LDA Color | Get color. |
| 13 | 9D | 00 | 20 | | STA(2000),x | Store according to x. |
| 16 | 8A | | | | TXA | Get index. |
| 17 | F0 | 04 | | | BEQ INTEN | Branch if finished. |
| 19 | CA | | | | DEX | Decrement x if not. |
| 1A | 4C | 11 | 03 | | JMP ALL1 | Do it again. |
| 1D | 20 | 7C | 03 | INTEN | JSR INPUT | Input from keyboard. |
| 20 | CA | 48 | | | CMP "H" | Is character an "H" |
| 22 | F0 | 05 | | | BEQ HIGH | Branch if yes. |
| 24 | A9 | 00 | | | LDA 00 | Prepare for low intensity. |
| 26 | 4C | 2B | 03 | | JMP LOW | Go around high. |
| 29 | A9 | 08 | | HIGH | LDA #$08 | Set high intensity bit. |
| 2B | 85 | 00 | | LOW | STA Color | Save intensity. |
| 2D | 20 | 7C | 03 | SHADE | JSR INPUT | Get shade. |
| 30 | 29 | 07 | | | AND #$07 | Change ASCII to binary. |
| 32 | 05 | 00 | | | OR Color | Combine shade and intensity. |
| 34 | 85 | 00 | | | STA Color | Save color (right byte) |
| 36 | 0A | | | ASL | | |
| 37 | 0A | | | ASL | | Form color in |
| 38 | 0A | | | ASL | | left byte. |
| 39 | 0A | | | ASL | | |
| 3A | 05 | 00 | | | ORA Color | Combine left and right. |
| 3C | 85 | 00 | | | STA Color | True full color. |
| 3E | 9D | 00 | 20 | STORE | STA(2000),x | Puts color on display. |
| ; | Decodes next command. | | | | | |
| 41 | 20 | 7C | 03 | DECODE | JSR INPUT | Get command. |
| 44 | C9 | 4E | | | CMP "N" | Is it an "N" |
| 46 | D0 | 03 | | | BNE NEXTA | Branch if not |
| 48 | 4C | 1C | 03 | | JMP INTEN | Prepare for new color. |
| 4B | C9 | 41 | | NEXTA | CMP "A" | Is it an "A?" |
| 4D | D0 | 03 | | | BNE NEXTU | Branch if not. |
| 4F | 4C | 0F | 03 | | JMP ALL | Jump if yes. |
| 52 | C9 | 55 | | NEXTU | CMP U | Is it a "U?" |
| 54 | D0 | 08 | | | BNE NEXTD | Branch if not. |
| 56 | 8A | | | | TXA | Prepare to change pointer. |
| ; | Move pointer vertically | | | | | |
| 57 | 38 | | | | SEC | Prepare to subtract. |
| 58 | E9 | 10 | | | SBC #16 | Subtract |
| 5A | AA | | | | TAX | Restore pointer. |
| 5B | 4C | 77 | 03 | | JMP CONTIN | Continue |
| 5E | C9 | 44 | | NEXTD | CMP "D" | Is it a "D?" |
| 60 | D0 | 08 | | | BNE NEXTL | Branch if not. |
| 62 | 8A | | | | TXA | Prepare to change pointer. |
| 63 | 18 | | | | CLC | Prepare to add. |
| 64 | 69 | 10 | | | ADC #16 | ADD |
| 66 | AA | | | | TAX | Restore pointer. |
| 67 | 4C | 77 | 03 | | JMP CONTIN | Continue |
| 6A | C9 | 4C | | NEXTL | CMP "L" | Is it an "L?" |
| 6C | D0 | 04 | | | BNE NEXTR | Branch if not. |
| 6E | CA | | | | DEX | Modify pointer. |
| 6F | 4C | 77 | 03 | | JMP CONTIN | Continue |
| 72 | C9 | 52 | | NEXTR | CMP "R" | Is it an "R?" |
| 74 | D0 | 01 | | | BNE CONTIN | Branch if not. |
| 76 | E8 | | | | INX | Modify pointer. |
| 77 | A5 | 00 | | CONTIN | LDA Color | Get color. |
| 79 | 4C | 3E | 03 | | JMP STORE | Start over. |
| ; | Subroutine to get character and turn Dazzler back on | | | | | |
| 7C | 20 | 5A | 1E | INPUT | JSR GETCHAR | |
| 7F | 85 | 01 | | | STA TEMP | Save accumulator. |
| 81 | A9 | 90 | | | LDA #$90 | |
| | | | | | | Turns Dazzler on. |
| 83 | 8D | 0E | 80 | | STA DAZZ | |
| 86 | A5 | 01 | | | LDA TEMP | Restores accumulator. |
| 88 | 60 | | | | RTS | |

*Program listing.*

UART card be removed from the mainframe.

Since the SWTP keyboard does not have a rubout (DEL) key, one will have to be added. The rubout signal is used by KIM to measure the incoming baud rate.

The keyboard has an unused key in the bottom right-hand corner. If the two terminals of this key are connected to lines Y-10 and X-3 (refer to your keyboard schematic), it will become a rubout.

The instructions for the keyboard also indicate that a jumper be used to select uppercase or uppercase and lowercase letters. Auxiliary switch 1 on the KIM System front panel can be used for this so that either mode can be easily selected.

**Some Demo Software**

Fig. 3 is the flowchart of a simple program that will enable you to "draw" on the top half of your TV screen using the keyboard. The program is intended only to demonstrate some of the KIM-1 System peripherals. Study the flowchart and try to rewrite the program to utilize the entire screen.

To load the program, set auxiliary switch 3 low so that the hex keypad is operational. Once loaded, set the address to 0300 and set auxiliary switch 3 to the high position. This will cause KIM to enter the TTY mode. Set the Print power switch high so that the printer will be enabled. Press RST and Rubout. KIM should then respond on the printer with the letters KIM.

Set the Dazzler power switch high and type a capital G to begin execution of the program. Type H or L for high or low intensity and then a number between 0 and 7 to select the color. The screen should show that color in the upper left-hand corner. Typing U, D, L or R will move the color up, down, left or right, respectively, drawing a line as it moves.

Typing A colors *all* the screen to the present color and returns the present position to the lower right-hand corner.

Typing N prepares the program to accept a new intensity and color.

A thorough study of the program should help the reader to utilize the keyboard and the Dazzler with his own programs.

Note that if the Print power switch is turned off, everything still functions properly, and the hard-copy output is eliminated.

**External Display and Hex Keyboard**

The remainder of this article deals with adding an external display and hex keyboard to KIM. Many people may feel that since the ASCII keyboard also provides the ability to load hex numbers, this step would really not be necessary.

Remember, however, that the hex keyboard has several additional functions such as RST, NMI interrupt and single step. Also, I find it much easier to use than the ASCII keyboard if I'm entering only hex data. Finally, there are many games available that utilize the KIM keyboard and display.

When I first looked through the KIM manual, I was glad to see that MOS had made provisions for an external keyboard through the application connector.

A little study will show, however, that a keyboard connected in that way will lack the three functions described above.

I decided to add three sockets to KIM and interface my external keyboard and display through them. The sockets were mounted by drilling holes with a #60 (no larger) drill bit for the pins. The wire was crimped around the solder tails of each pin. This connection, when

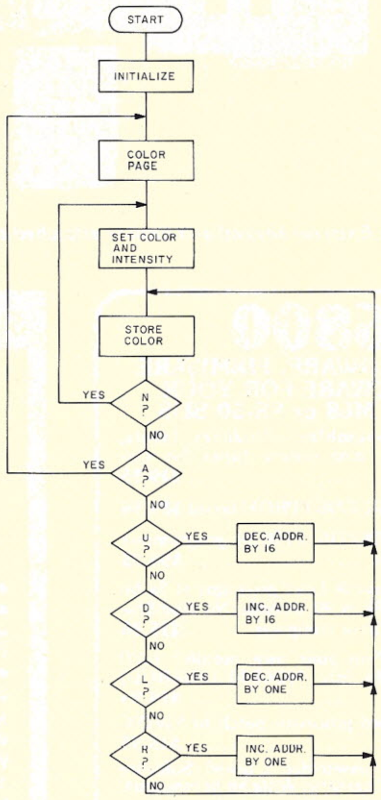



*Fig. 3. Program flowchart.*

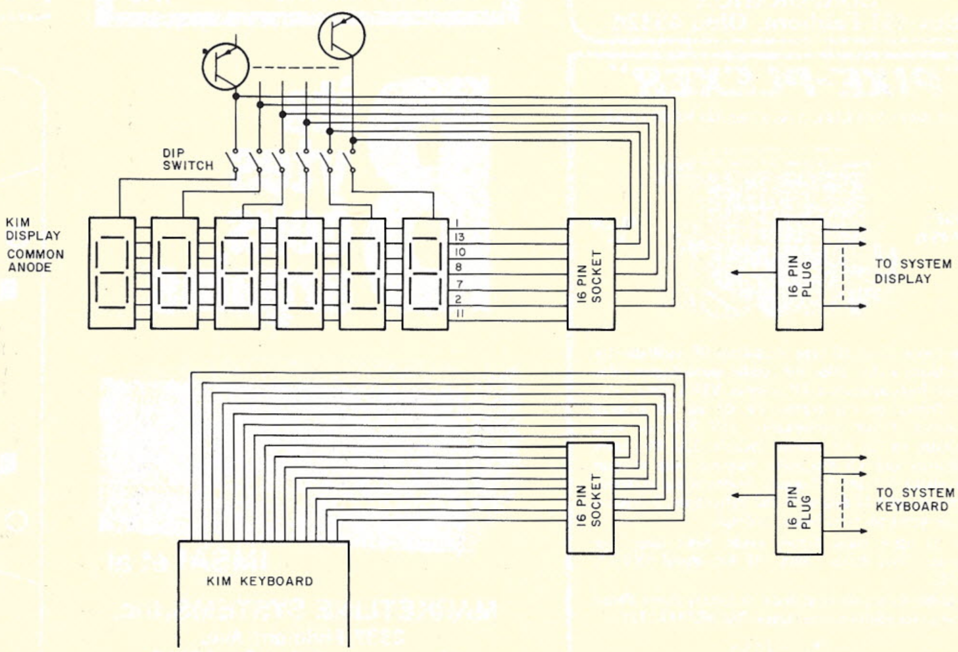



*Fig. 4. Keypad and external display schematic for KIM.*

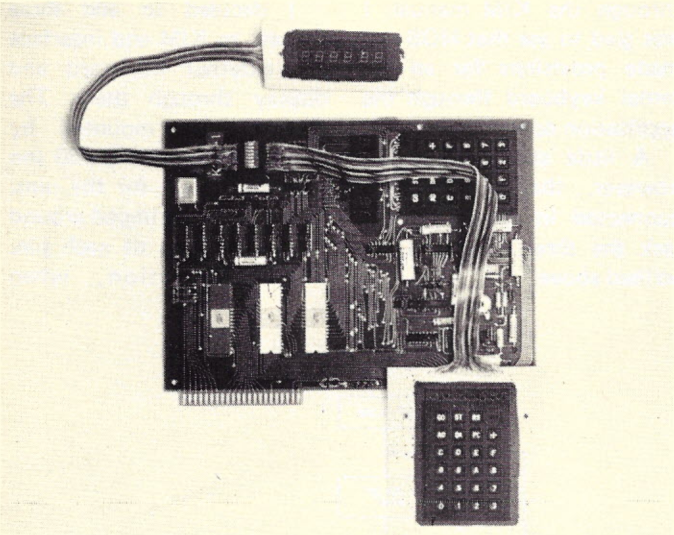


*Photo 2. External keypad and display attatched to KIM.*

soldered, holds the sockets tightly in place.

The keyboard is mounted on a printed circuit board to make the wiring and mounting easier. The cable wiring should simply connect the pins of one keyboard to the *same* pin on the other keyboard. A 16-pin DIP plug on one end of the cable allows the keyboard to be easily disconnected from KIM.

The display could be handled in the same way, but I chose to cut the foil lines on KIM that provided the multiplexed power lines to the LEDs so that only the external display is functional. A six-switch DIP is used to enable the on-board display if I should need it in the future. My external LEDs are mounted in wire-wrap sockets on Vector board. The board is painted black so that it is less noticeable when mounted. Fig. 4 is a schematic layout of the connections, and Photo 2 shows the modified KIM.

Most of your front panel is now operational; the next article on the external interface board will complete the system's hardware, bringing to life the sense switches, joysticks and D/A ports. ■

*Mike L. Simon*
*Space-Time Productions*
*2053 N. Sheffield*
*Chicago IL 60614*

# Faster Erase Times

## *build a quicker EPROM eraser*

The price of EPROMs has been dropping rather drastically lately. Still, it is not low enough to allow throwing one away just to change the program; hence the need for an EPROM eraser.

The first EPROM eraser I built cost about $8 for parts. It used a GE G4S11 bulb in series with a 40 Watt light bulb and operated off the ac line (see *Byte*, January 1977, p. 91). This worked, but since it took approximately 30 minutes to give an erased indication, total erase time was two hours. Intel recommends erasing for three to four times the minimum erasure time.

I considered buying the Byte Destroyer (*Kilobaud*, June 1977, p. 65) but decided I could have some fun and save money by building another eraser myself.

### Construction

The GE germicidal lamp catalog lists the G4T4/1, which, with the proper ballast, produces almost eight times the power output of the G4S11. That means full erasure in 15 to 20 minutes, certainly a time saver.

The lamp, socket and ballast cost about $18, or can be ordered from Space-Time Productions, 2053 N. Sheffield, Chicago IL 60614, for $22 postpaid. The housing is a bread pan 8½ x 4½ x 2½ inches (see Fig. 1). The socket is an Amphenol 77MIP4 or equivalent (4-pin tube socket), and the ballast is a GE 89G435.

I got the bread pan from a local grocery store. The bulb, socket and ballast came from local suppliers, and the push-button switch was in the junk box. A grounded cord is a must since the parts are all mounted on an uninsulated metal chassis (the bread pan) as shown in Fig. 1.

Since the bottomless junk box also provided a mounting bracket for the socket, I did not use the bracket shown in Fig. 2. However, it should be easy to make with simple tools (pliers, hacksaw and hammer, if necessary!). Place the bracket so that the bulb is 1¼ inch from the surface the EPROMs rest on; this will locate them one inch from the surface of the bulb.

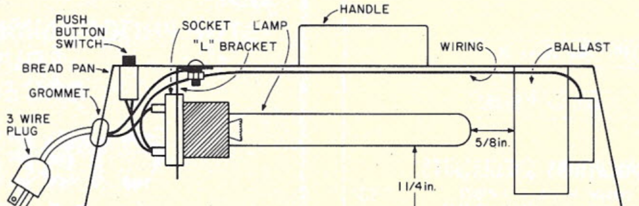



*Fig. 1. EPROM eraser.*

Position the components in the pan and mark the holes for drilling. Be sure to allow 5/8 inch between the end of the bulb and the ballast. This allows for easy removal of the bulb. Then fasten everything together, wire it and fire it up. Works the first time! It's either dumb luck, or this project is too simple to mess up. (Checking the wiring diagram in Fig. 3, I concluded the latter.)

Starting procedure is simple. Hold the starting switch for one second, then let go. Almost instantaneous.

For those who prefer more automatic operation, an FS-5 starter may be substituted for the switch. For the ultimate unit, wire the switch and starter in parallel. Then, if it doesn't work in automatic, you have manual in reserve.

If you want less light output and probably more filament life, the GE 58G827-60 cycle ballast will reduce lamp output by 40 percent. Exposure times must be increased by 70 percent with this ballast – from 20 to 35 minutes erasure time. That is probably about the same time required for the G8T5 lamp (used in the Byte Destroyer), since GE rates both lamps approximately equivalent in light output at close distances when the G4T4/1 lamp is used with the 58G827 ballast. I haven't tried this, so I can't say for sure.

A handle for the top of the chassis makes it easier to lift, especially considering the smooth sloping sides. I made a base for the unit that helps put the EPROMs under the lamp. I marked the outline of the chassis on the base, measured the area under the lamp and then marked it on the base.

### EPROM Theory

The 2708 is a floating gate MOS EPROM (see Fig. 4). With no charge on the gate, the resistance between drain and source is very high, giving a one output when accessed. Twenty-six volts between the drain and source causes an avalanche of electrons from the substrate. Some of these electrons have enough energy to penetrate the silicon dioxide ($SiO_2$) insulation between the channel of the FET and the floating gate.

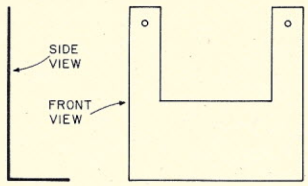



*Fig. 2. L bracket.*

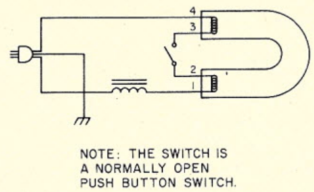



*Fig. 3. Schematic diagram of eraser.*

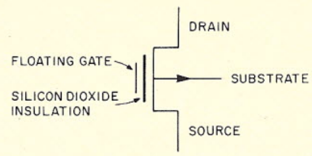



*Fig. 4. Floating gate MOSFET.*

Charges accumulated on the gate during programming cause the resistance between the drain and source to be much lower, giving a zero output when accessed. Since EPROMs are MOS devices and are subject to being blown by static electricity, you must short all leads together whenever the EPROM is not in a socket on a computer board. You can use aluminum foil or conductive foam.

Proper programming requires that each byte be programmed with pulses of .1 to one millisecond long for a total program time of 100 ms per byte. Each byte must be programmed in sequence, starting from the zero byte with none left out. The 2708 can be reprogrammed without erasing as follows: The first time you program an EPROM, all "don't care" bits must be programmed as ones. To reprogram ones to zeros, you must also reprogram all other bits previously programmed. If any zero bits must be programmed to ones, the EPROM must be erased and reprogrammed.

**8080/2708 Programming**

I have a Byte Saver from Cromemco that programs each byte for .3 ms. I have written Program A for the 8080 that programs each byte for a total of approximately 150 ms. This gives a 50 percent safety factor over what Intel recommends.

To erase the EPROM, the charge must be removed from the floating gate. To do this, a source of high-intensity shortwave ultraviolet light is needed. This light shines through a thin quartz window provided on the EPROM (which *must be clean*). The light absorbed by the electrons on the gate gives them enough added energy so they can again penetrate the insulating layer and leak off into the drain source and substrate.

Shortwave ultraviolet fades colors and can also fade eyes. *Do not look at any ultraviolet source without eye protection*! — particularly shortwave lamps, as their rays are especially damaging. Face shields can be obtained from safety equipment manufacturers. Arc-welding goggles would probably prove satisfactory.

Mercury vapor gives off intense radiation at 253.7 nanometers wavelength when it is ionized. This light is effective in killing bacteria, which is why GE calls the bulbs germicidal lamps. The tube resembles a regular fluorescent except that it has no phosphor and the glass is a special type that transmits the 253.7-nanometer wavelengths. This particular tube is bent in a U shape instead of being straight like most fluorescent tubes. As the tubes are used, the ultraviolet output power drops — by about 25 percent during the first 100 hours. Over the next three to five thousand hours, the output drops 25 percent below the 100-hour output. Keep this in mind when erasing EPROMs after initially calibrating the setup.

In a fluorescent tube of this kind, when the switch is closed, current flows through the ballast, building up a magnetic field around the ballast and storing energy in the field. This current flows through the filaments in the bulb, vaporizing the mercury and lowering the ionization voltage of the bulb. When the switch is released, the collapse of the magnetic field around the ballast induces a voltage in the ballast that causes the lamp to ionize and start burning. Since the lamp will not limit the current it draws as an incandescent lamp will, the ballast limits the current through the lamp.

**Wrap-up**

The setup can be calibrated by writing an EPROM and then erasing it for an interval and testing for all bytes erased. Multiplying the minimum erasure time by four gives the total erasure time.

Some EPROM programmers are designed to give a programming time of five times the minimum write time (*PROM Users Guide*, Pro Log, Monterey CA 93940). Since the time taken to program all bytes in different EPROMs varies, this can save programming time. Most important, it gives confidence that each gate has been given the minimum amount of charge necessary for reliable operation. It is possible for some EPROMs to require more than 100 ms per byte for a minimum charge.

I did not do this in the program given here since I wanted a program that would get my system up quickly with a minimum of bootstrapping. Later, as I get more of my system together, I plan to add an interactive EPROM program.

By now you have acquired sufficient information to use the tools. Happy PROMing! ■

| Address | Code | Mnemonics | Remarks |
|---|---|---|---|
| 000 | 006 | MVI B | Zero burn counter. |
| 001 | 000 | 000 | |
| 002 | 041 | LXI H | Set source pointer. |
| 003 | 000 | 000 | |
| 004 | 010 | 010 | |
| 005 | 021 | LXI D | Set destination pointer. |
| 006 | 000 | 000 | |
| 007 | 274 | 274 | |
| 010 | 176 | MVI AM | Move a block. |
| 011 | 022 | STA D | |
| 012 | 043 | INX H | |
| 013 | 023 | INX D | |
| 014 | 172 | MOV AD | |
| 015 | 376 | CPI | Test for end of block. |
| 016 | 300 | 300 | |
| 017 | 302 | JNZ | If not end, move another byte. |
| 020 | 010 | 010 | |
| 021 | 000 | 000 | |
| 022 | 004 | INR B | If end of block, increment burn counter. Test for end of burn. |
| 023 | 302 | JNZ | If not end, burn another block. |
| 024 | 002 | 002 | |
| 025 | 000 | 000 | |
| 026 | 166 | HALT | END |

*Program A. PROM burner program written for Cromemco Byte Saver with PROM to be burned located at 274 000. Source of data to be written starts at 010 000.*

# I/O Programming for the Altair Disks

## *ain't no big deal*

*Don Alexander*
*Microcomputer Ventures Inc.*
*4497 Indianola Ave.*
*Columbus OH 43214*

This article is being written at the prompting of several people who own or have used the Altair disk units and wish to know how to do I/O without BASIC. There are some peculiarities about the drive which are not well known, and require software tricks which complicate matters when one tries to write an I/O routine from the information presented in the operating manual. So here is a general purpose I/O routine for the Altair disk. But first, a short history of my initial trauma with the disk for comic relief.

I acquired my disk in the spring of 1976 when I had the good fortune of winning it at the first World Altair Computer Convention. What I didn't win was software support for the disk.

Assuming the lack of software to be an oversight as I knew from the Mits advertisements that a disk operating system (DOS) was supplied with their disk drives, I proceeded to make several fruitless calls to Mits. But there was no DOS.

I was still poor then (not having begun to write for *73/Kilobaud*) so I was not willing to shed the bucks for BASIC. Besides, I was more interested in real-time applications for which BASIC is not suited. So I pulled out the thick manuals accompanying the disk drive and sat down to write my own disk I/O routines which I could later use in my DOS. I found only a short preliminary user's section buried in reams of construction notes. But the final version of this section was supposed to be out by then so I once again called Mits. Well, it wasn't ready. In fact, it may not have been updated yet for all I know. I was supposed to receive the updated manuals as soon as they were printed, but have yet to get anything.

Back to the drawing board. I wrote some simple I/O routines based on the information at hand. Surprise – they worked. Sometimes.

A day later, almost at wit's end, I discovered that the execution time for the program loop required to write (or read) data was in the range of 30-35 microseconds, depending on whether a conditional jump after a test was taken. But successive bytes of data are written every 32 microseconds. I stared at the program for hours and could see no way to shorten the loop.

More phone calls. Eventually I was put in touch with the engineer in charge of the disk project. I had talked to him before and had been told that the manuals had enough information to write the software. This time, since I was able to say for sure that the simplest program loop was too long, he acknowledged that there was a software trick which had to be used to get around that problem and outlined the procedure to me.

It's not so much of a software trick as it is a replacement of hardware by software. One merely makes the read/write loops handle two bytes per revolution (of the loop) where the second byte is timed by software rather than using the sync lines, as is more pleasing. I'll explain how this works later in the software description.

During the course of my conversations with Mits personnel I learned various other facts about the disk drive, one which strikes me as particularly amusing. It

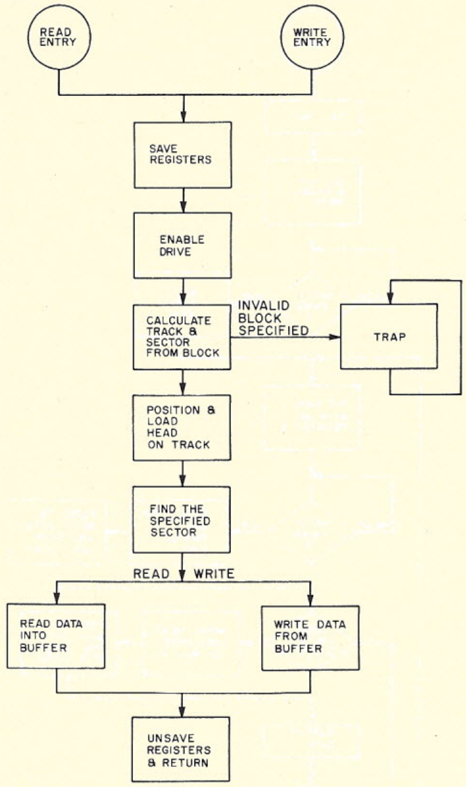



*Fig. 1. Simplified flowchart showing major sections of code.*

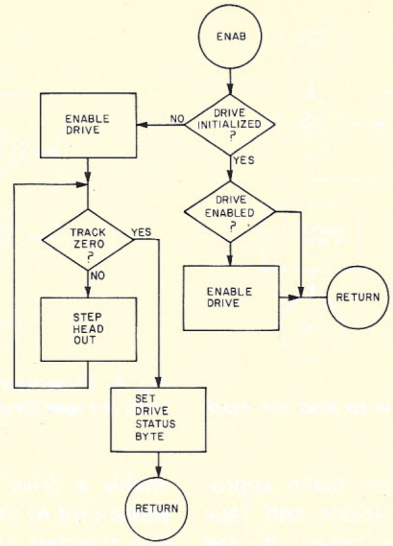



*Fig. 2. Flowchart of the disk enable routine called ENAB.*

concerns the use of interrupts with the disk.

While no interrupts are allowed during actual transfer of data (this would mess up the already critical timing) it is possible to use interrupts while trying to find a specific sector on the disk. If enabled to generate interrupts, the controller will interrupt the CPU as the beginning of each sector passes under the read/write head of the drive. The computer is then supposed to check to see if this is the sector it is looking for and if so, do I/O. The thing that struck me as peculiar was that interrupt level 7 is used for this purpose. Level 7 is the lowest priority interrupt level and there is not too much time to waste once the beginning of the sector passes under the head. What I discovered in talking with the engineers was that they believed level 7 was the highest priority interrupt. Guess it's just as hard to obtain accurate information on the inside as it is on the outside. My routines don't use the interrupt, so it was no problem to me.

Armed with the information finally provided by the engineers, I was able to make my disk routines work. In fact, they have worked very reliably. Though I would have preferred the disk controller to be a bit more intelligent, the Altair disk system works quite well.

## Using This I/O Routine

The basic flow of the disk I/O program is shown in Fig. 1. I want to emphasize that this is only an I/O routine and does not constitute anything close to a DOS. It merely provides the capability to read and write to the disk.

The calling program (any program you write that makes use of the disk routine presented here) is required to put the starting address of the memory block to be written to disk in the HL register pair. For a read operation the HL register should contain the starting address where the data on disk is to be placed in memory. Register A should contain the total number of bytes to be written or read (up to 136). Register B should contain the drive number. If you have only one drive, then this would always be zero and you could modify the disk routine so the calling program would not have to specify the disk drive. Register pair DE should contain the block number to be used. There are 77 tracks of 32 sectors each on the disk for a total of 2464 sectors. The block number is simply a number from 0-2463 to specify which sector to use. This is easier than making the calling routine specify the track and sector when you are outputting large blocks of data that may fill several tracks. Once these registers have been set up, entry to the disk routine is made by a CALL to either READ or WRITE. The disk routine will perform the desired operation and return to the calling program with the registers unchanged.

This routine does not attempt to do any error checking or special formatting. I chose to leave the error checking to the calling program as there are several schemes which may be employed (I'm told that format and error checking varies between one version of BASIC and the next, though I don't know for sure). In any case the I/O routine would not do error checking during actual data transfer as there is no time for that. If you wish to use checksums or other devious schemes to detect errors, you should move data

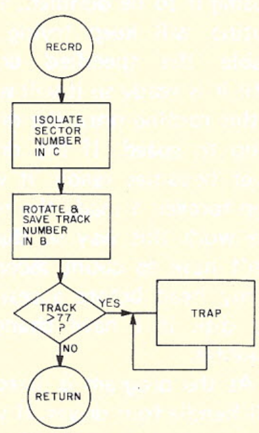



*Fig 3. Flowchart of the routine RECRD, which extracts the sector and track numbers from the block number.*

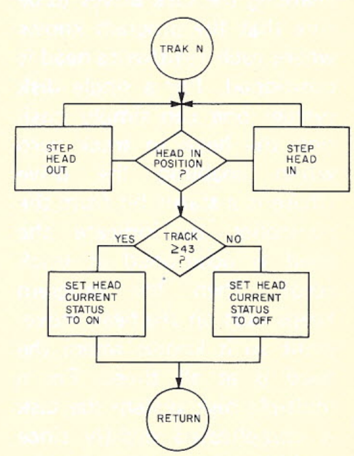



*Fig. 4. Flowchart of TRKN, which positions the head and determines status of the head current switch.*

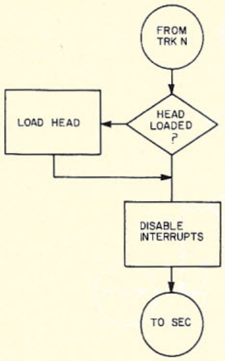



*Fig. 5. Routine to load the head.*

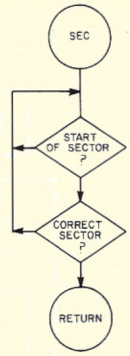



*Fig. 6. Flowchart of SEC, which finds the specified sector.*

into a buffer, make appropriate calculations and tack the necessary bytes onto the buffer before calling this disk routine to write it to disk. When you read it back, do the same in reverse.

The following is a description of the various subroutines which make up my I/O routine.

**Saving Registers**

The first thing that either the READ or WRITE routines do is to call the subroutine INIT which saves registers and calls other subroutines to set up the disk drive. Before actually saving registers INIT pulls the return address off the stack so it can be put back on after the registers have been saved. When the disk is ready for I/O it will then return to the right place.

**Enabling the Disk Drive**

Care must be taken when enabling the disk drives to be sure that the program knows where each read/write head is positioned. For a single disk system one can simply position the head to track zero when enabling the drive (there is a status bit from the controller to indicate the head is positioned at track zero). Then the program keeps tabs on the head movement so it knows where the head is at all times. For a multiple disk system the task is complicated slightly since enabling one drive disables the others. You wouldn't want to position a head to track zero each time you enable a drive as this would waste a lot of time on disk-to-disk transfers, so you have to keep track of the position of all disk heads. But the first time you enable a drive after the system is turned on you must be sure to move its head to track zero to provide a reference.

This is handled by the routine called ENAB. It first checks to see if the drive specified in register B has been initialized by pulling a status byte for that drive from a table. If it has not been initialized, a branch to the routine TRK0 is taken which enables the drive and positions its head to track zero. If it has been initialized, a check is made to determine if the drive which is currently in use is the same as is requested. If not, the new drive is enabled. The routine also checks to be sure that the specified drive is really enabled as the operator may have opened the drive door causing it to be disabled. The routine will keep trying to enable the specified drive until it is ready so it will wait in this routine until the drive is up to speed. If the drive never becomes ready, it will loop forever. I made the routine work this way so that I don't have to count seconds in my head before accessing the disk if I have changed diskettes.

As the program is listed it will handle four drives. If you have more than four drives, you need only add more bytes to the table beginning at TRKNM. You should also

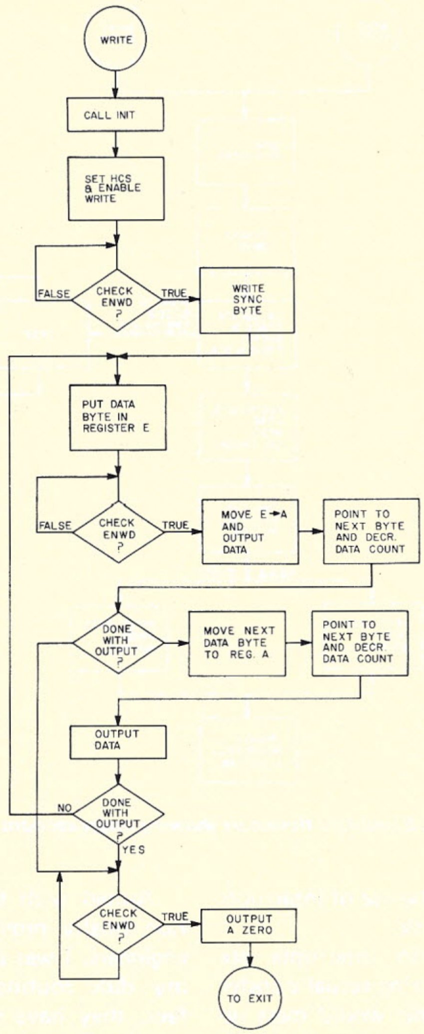



*Fig. 7. Flowchart of the WRITE routine.*

guarantee that the status bytes in this table are all set to 80 hex when you first load your system so the enable routine will know to initialize the disk drives. A simple flowchart of the enable routine is shown in Fig. 2.

**Calculating Track and Sector Numbers**

It is more convenient to pass a two-byte block number to specify the desired sector than to pass both a track and sector number as separate bytes. But positioning the head requires the track and sector numbers to be separated. This task is handled by the subroutine RECRD (flowchart in Fig. 3). It is very simple as the low five bits of the DE register pair are the sector number and the next seven are the track number. RECRD isolates the sector number in register C and normalizes and isolates the track number in register B. RECRD also checks to be certain that a valid track is specified. If not, it will jump to a trap location which loops on itself, thus bombing the system. I did this because I found that the disk drive makes a horrendous crunching sound if you attempt to do a read spindle operation. You could be a little classier by jumping to a routine to clear the stack, print an error message and abort the program trying to use the disk. Note that I have left out a test for specification of a valid disk drive as the program will loop forever if that happens anyway. The check made here is intended to prevent destruction of hardware by the software.

**Positioning the Head**

Head positioning is done

by the subroutine TRKN (flowchart in Fig. 4). The status byte from the table of status bytes contains the current head position for the disk drives and is used to determine if the head should be stepped in or out. Once the head is on the correct track, TRKN sets a status byte which will later be used by the write routine to enable its write channel and to set what is called the head current switch (HCS). The HCS is a hardware feature to reduce the write current passing through the disk head to allow for better resolution on higher tracks. TRKN will set the status byte to turn on the HCS if the head is positioned on or above track 43.

### Loading the Head

Loading the head refers to placing the read/write head in contact with the surface of the diskette. Normally it is held away from the disk to increase both head and disk life. Loading the head is flowcharted in Fig. 5. The reason for testing to see if the head is loaded (rather than simply issuing the load command) is that a load command always triggers a delay internal to the disk controller. This delay prevents I/O until the head has settled. If the load command were always issued, it would be impossible to write data to successive sectors on the diskette during the same revolution of the disk because of the delay.

For the same reason I have elected not to unload the head anywhere in the disk routine presented here. That task should be performed by the calling program when it is finished with all the I/O it intends to do for the time being. This allows the head to remain loaded if you are in the process of writing multiple sectors to the disk. Unloading the head is accomplished by outputting an 8 to port number 9.

*Program listing.*

```
0000                  ;DISK HANDLING ROUTINE FOR THE ALTAIR DISK DRIVE
0000                  ;
0000                  ;ENTER AT READ OR WRITE WITH THE FOLLOWING VALUES
0000                  ;IN REGISTERS A,B,DE,AND HL
0000                  ;
0000                  ; REG A CONTAINS NUMBER OF BYTES TO BE INPUT OR OUTPUT
0000                  ; REG B CONTAINS THE DISK DRIVE NUMBER
0000                  ; REG DE CONTAINS THE BLOCK NUMBER RANGING FROM 0
0000                  ;       TO 2463 DECIMAL
0000                  ; REG HL CONTAINS THE BUFFER ADDRESS WHERE DATA IS TO
0000                  ;       BE READ FROM OR WRITTEN TO
0000                  ;
0000                  ;ALL REGISTERS ARE RETURNED INTACT
0000                  ;INTERRUPTS ARE NOT ALLOWED DURING DISK I/O AND THE
0000                  ;INTERRUPTS ARE ENABLED ON RETURN
0000                  ;
0000                  ;
0000                  ;SUBROUTINE TO ENABLE DISK DRIVE
0000                  ;
0000  21  40  01      ENAB LXI H,TRKNM ;POINT TO START OF TRACK TABLE
0003  78               MOV A,B ;PUT DRIVE NUMBER IN REG A
0004  3D              ENAB1 DCR A ;INCREMENT HL SO THAT IT WILL
0005  FA  0C  00       JM ENAB2 ;POINT TO THE CORRECT STATUS BYTE
0008  23               INX H
0009  C3  04  00       JMP ENAB1
000C  7E              ENAB2 MOV A,M ;GET STATUS BYTE FOR SPECIFIED DRIVE
000D  E6  80           ANI 80H ;TEST IF IT HAS BEEN INITIALIZED
000F  C2  2C  00       JNZ TRKO ;IF NOT THEN INITIALIZE IT
0012  3A  3F  01       LDA DRVNM ;GET CURRENT DRIVE NUMBER
0015  B8               CMP B ;IS IT THE SAME AS REQUESTED DRIVE?
0016  C2  1E  00       JNZ ENAB3 ;IF NOT THEN ENABLE NEW DRIVE
0019  DB  08           IN DSTAT ;CHECK IF CURRENT DRIVE IS ENABLED
001B                  ;IN CASE DRIVE DOOR HAS BEEN OPENED
001B  E6  08           ANI ENBIT
001D  C8               RZ RETURN IF ENABLED
001E  78              ENAB3 MOV A,B ;GET DRIVE NUMBER
001F  32  3F  01       STA DRVNM ;AND SAVE IT
0022  D3  08           OUT DSTAT ;ENABLE DRIVE
0024  DB  08           IN DSTAT ;IS DRIVE ENABLED NOW?
0026  E6  08           ANI ENBIT
0028  C2  1E  00       JNZ ENAB3 ;IF NOT KEEP TRYING
002B  C9               RET
002C                  ;
002C                  ;SUBROUTINE TO FIND TRACK 0
002C                  ;
002C  CD  1E  00      TRKO CALL ENAB3 ;ENABLE DISK DRIVE
002F  CD  95  00      TRK01 CALL MOVE ;BE SURE HEAD MOVE IS ALLOWED
0032                  ;       TO GUARANTEE HEAD IS SETTLED
0032  DB  08           IN DSTAT ;GET DISK STATUS
0034  E6  40           ANI TOBIT ;CHECK FOR TRAK ZERO
0036  CA  3F  00       JZ TRK02 ;GO SET STATUS IF ON TRACK 0
0039  CD  40  00       CALL OUT ;IF NOT 0, STEP OUT
003C  C3  2F  00       JMP TRK01 ;CHECK AGAIN FOR TRACK 0
003F  77              TRK02 MOV M,A ;SET DISK STATUS BYTE TO INDICATE
0040                  ;ZERO AND THAT DRIVE HAS BEEN INITIALIZED
0040                  ;
0040                  ;SUBROUTINE TO STEP HEAD OUT
0040                  ;
0040  CD  95  00      OUT CALL MOVE ;BE SURE HEAD MOVE IS ALLOWED
0043  3E  02           MVI A,OUTBT ;GET STEP OUT COMMAND
0045  D3  09           OUT DCONT ;STEP HEAD OUT
0047  C9               RET
0048                  ;
0048                  ;SUBROUTINE TO STEP HEAD IN
0048                  ;
0048  CD  95  00      IN CALL MOVE ;BE SURE HEAD MOVE IS ALLOWED
004B  3E  01           MVI A,INBIT ;GET STEP IN COMMAND
004D  D3  09           OUT DCONT ;STEP HEAD IN
004F  C9               RET
0050                  ;
0050                  ;SUBROUTINE TO FIND TRACK N AND SET THE
0050                  ;HEAD CURRENT SWITCH STATUS
0050                  ;
0050  7E              TRKN MOV A,M ;GET CURRENT TRACK NUMBER
0051  B8               CMP B ;CHECK FOR DESIRED TRACK
0052  CA  68  00       JZ STHCS ;IF EQUAL GO SET HEAD CURRENT SWITCH
0055  DA  60  00       JC MVIN ;STEP HEAD IN IF B IS GREATER THAN A
0058  3D              MVOUT DCR A ;DECREMENT TRACK NUMBER
0059  77               MOV M,A ;AND SAVE IT
005A  CD  40  00       CALL OUT ;AND STEP HEAD OUT
005D  C3  50  00       JMP TRKN ;CHECK AGAIN FOR CORRECT TRACK
0060  3C              MVIN INR A ;INCREMENT TRACK NUMBER
0061  77               MOV M,A ;AND SAVE IT
0062  CD  48  00       CALL IN ;AND STEP HEAD IN
0065  C3  50  00       JMP TRKN ;CHECK AGAIN FOR CORRECT TRACK
0068  FE  2B          STHCS CPI 43 ;IS TRACK GT OR EQ TO 43?
006A  3E  C0           MVI A,HCSON ;GET HEAD CURRENT ON COMMAND
006C  D2  71  00       JNC STHC ;IF GT OR EQ TO 43 SET HEAD CURRENT SWITCH
006F  3E  80           MVI A,HCOFF ;GET HEAD CURRENT OFF COMMAND IF LT 43
0071  32  44  01      STHC STA HCS ;SAVE HEAD CURRENT SWITCH STATUS
0074  C9               RET
0075                  ;
0075                  ;SUBROUTINE TO DEDUCE TRACK AND SECTOR NUMBERS
0075                  ;FROM LOGICAL RECORD NUMBER SUPPLIED BY CALLING ROUTINE
0075                  ;
0075  7B              RECRD MOV A,E ;GET LOW BYTE OF RECORD NUM
0076  E6  1F           ANI 1FH ;ISOLATE LOW FIVE BITS (SECTOR ADDR)
0078  4F               MOV C,A ;SAVE SECTOR ADDR IN C
0079  7B               MOV A,E ;GET LOW BYTE AGAIN
007A  CD  91  00       CALL ROT ;ROTATE TO GET LOW THREE BITS OF TRACK NUM
007D  E6  07           ANI 07H ;ISOLATE THOSE THREE DATA BITS
007F  47               MOV B,A ;SAVE THEM IN B
0080  7A               MOV A,D ;GET HIGH BYTE OF RECORD NUM
```

```
0081   CD   91   00    CALL ROT ;JUSTIFY IT AS ABOVE
0084   E6   F8         ANI 0F8H ;ZERO LOW THREE BITS
0086   B0              ORA B ;OR IN THE LOW THREE BITS
0087   FE   4E         CPI 78 ;CHECK FOR VALID TRACK
0089   D2   3C   01    JNC ERR1 ;JUMP IF ERROR
008C   47              MOV B,A ;SAVE TRACK NUMBER IN B
008D   32   48   01    STA TEMP
0090   C9              RET
0091                  ;
0091                  ;SUBROUTINE TO ROTATE THREE TIMES
0091                  ;9
0091   07              ROT RLC
0092   07              RLC
0093   07              RLC
0094   C9              RET
0095                  ;
0095                  ;SUBROUTINE TO WAIT TILL HEAD MOVE IS ALLOWED
0095                  ;
0095   DB   08        MOVE IN DSTAT ;GET DISK STATUS
0097   E6   02         ANI MVBIT ;CHEC HEAD MOVE BIT
0099   C2   95   00    JNZ MOVE ;WAIT TILL MOVE IS ALLOWED
009C   C9              RET
009D                  ;
009D                  ;SUBROUTINE TO SAVE REGISTERS AND INVOKE SUPPORT ROUTINES
009D                  ;TO PREPARE DISK FOR READ OR WRITE
009D                  ;
009D   22   46   01   INIT SHLD BUFAD ;SAVE BUFFER ADDRESS
00A0   E3              XTHL ;PUSH HL AND GET RETURN ADDRESS
00A1   D5              PUSH D ;SAVE OTHER REGISTERS
00A2   C5              PUSH B
00A3   F5              PUSH PSW
00A4   E5              PUSH H ;PUT RETURN ADDRESS BACK ON STACK
00A5   32   45   01    STA BYTES ;SAVE BYTE COUNT
00A8   CD   00   00    CALL ENAB ;ENABLE DISK DRIVE
00AB   CD   75   00    CALL RECRD ;GET TRACK AND SECTOR NUMBERS
00AE   CD   50   00    CALL TRKN ;POSITION HEAD
00B1   2A   46   01    LHLD BUFAD ;GET BUFFER ADDRESS BACK IN HL
00B4   DB   08         IN DSTAT ;GET DISK STATUS
00B6   E6   04         ANI HDBIT ;CHECK IF HEAD IS LOADED
00B8   CA   BF   00    JZ OKAY ;SKIP LOAD IF ALREADY LOADED
00BB   3E   04         MVI A,HDBIT
00BD   D3   09         OUT DCONT ;LOAD HEAD
00BF   3A   45   01   OKAY LDA BYTES ;GET BYTE COUNT
00C2   57              MOV D,A ;AND KEEP IN D
00C3   F3              DI ;DON'T ALLOW INTERRUPTS DURING DISK I/O
00C4                  ;
00C4                  ;SUBROUTINE TO FIND START OF DESIRED SECTOR
00C4                  ;
00C4   DB   09        SEC IN POS ;READ SECTOR POSITION STATUS
00C6   47              MOV B,A ;SAVE IN B
00C7   E6   01         ANI SECBT ;TEST FOR START OF SECTOR
00C9   C2   C4   00    JNZ SEC ;IF NOT START TRY AGAIN
00CC   78              MOV A,B ;GET SETOR NUMBER IN ACCUM
00CD   1F              RAR ;JUSTIFY IT
00CE   E6   1F         ANI 1FH ;ISOLATE ADDRESS BITS
00D0   B9              CMP C ;CHECK FOR DESIRED SECTOR
00D1   C8              RZ ;RETURN TO READ OR WRITE IF CORRET SECTOR
00D2   C3   C4   00    JMP SEC ;IF NOT, TRY AGAIN
00D5                  ;
00D5                  ;ENTRY POINT TO READ A SECTOR
00D5                  ;
00D5   CD   9D   00   READ CALL INIT ;GET READY TO READ
00D8   DB   08        RSYN IN DSTAT ;GET DISK STATUS
00DA   17              RAL ;SHIFT DATA AVAILABLE BIT TO CARRY
00DB   DA   D8   00    JC RSYN ;LOOP TILL DATA IS READY
00DE   DB   0A         IN DATA ;READ SYNC BYTE
00E0   DB   08        FB IN DSTAT ;GET DISK STATUS
00E2   17              RAL ;SHIFT TO CARRY
00E3   DA   E0   00    JC FB ;KEEP LOOKING FOR FIRST BYTE
00E6   DB   0A         IN DATA ;READ FIRST BYTE
00E8   5F             RDAT1 MOV E,A ;PUT DATA IN E SO READ ROUTINE WILL WORK
00E9   DB   08        RDAT IN DSTAT ;GET DISK STATUS
00EB   17              RAL ;SHIFT TO CARRY
00EC   DA   E9   00    JC RDAT ;KEEP LOOKING FOR DATA READY
00EF   DB   0A         IN DATA ;READ DATA
00F1   73              MOV M,E ;STORE FIRST BYTE
00F2   23              INX H ;POINT TO NEXT BYTE IN BUFFER
00F3   15              DCR D ;DECREMENT BYTE COUNT
00F4   CA   FF   00    JZ EXIT ;EXIT IF DONE
00F7   77              MOV M,A ;IF NOT DONE, STORE THIS BYTE IN BUFFER
00F8   23              INX H ;POINT TO NEXT BYTE IN BUFFER
00F9   15              DCR D ;DECREMENT BYTE COUNT
00FA   DB   0A         IN DATA ;TIME TO READ NEXT BYTE FROM DISK
00FC   C2   E8   00    JNZ RDAT1 ;READ MORE IF NOT DONE
00FF                  ;
00FF                  ;ROUTINE TO LEAVE DISK HANDLER
00FF                  ;
00FF   F1             EXIT POP PSW ;RESTORE REGISTERS
0100   C1              POP B
0101   D1              POP D
0102   E1              POP H
0103   FB              EI ;ENABLE INTERRUPTS
0104   C9              RET ;GO BACK TO CALLING ROUTINE
0105                  ;
0105                  ;ENTRY POINT FOR WRITING TO DISK
0105                  ;
0105   CD   9D   00   WRITE CALL INIT ;GET READY TO WRITE
0108   3A   44   01    LDA HCS ;GET HEAD CURRENT SWITCH STATUS
010B   D3   09         OUT DCONT ;SET HEAD CURRENT AND WRITE ENABLE
010D   1E   FF         MVI E,SYNC ;GET SYNC BYTE IN E
010F   DB   08        WSYN IN DSTAT ;GET DISK STATUS
0111   1F              RAR ;SHIFT ENTER NEW WRITE DATA BIT TO CARRY POSITION
0112   DA   0F   01    JC WSYN ;KEEP LOOPING TILL READY
```

### Finding the Right Sector

The only thing left to do before I/O can begin is to locate the specified sector (flowchart in Fig. 6). The Altair disk controller provides a status bit to indicate when the beginning of a sector is passing under the head. When this status bit is true the program reads the sector address and checks it against the requested sector. If it matches, the program returns to either the read or write routines to begin I/O. Otherwise it keeps looking.

### Writing Data

The WRITE routine does the actual output of data as shown in Fig. 7. On return from the common section of code (all routines discussed above), the WRITE routine immediately enables the write electronics of the controller by outputting the status byte set up by the TRKN routine. When the Enter New Write Data (ENWD) status line goes true a sync byte is written. The sync byte is a hex FF and is kept in register E while testing ENWD so it can be transferred to the accumulator as quickly as possible. Then the first byte of data is loaded into register E and ENWD is again tested. When ENWD goes true the first byte is written and bookkeeping is performed.

The bookkeeping consists of incrementing the HL registers to point to the next data byte and decrementing the data count which at this point is in register D. If all data has been written, the program branches down to write a stop byte in a similar fashion. If not, the next byte of data is loaded into register A and bookkeeping happens again. At this point in the program enough time has elapsed since the last byte was written that it is now time to write another byte. The byte now in register A is output to disk. After this byte is written the test for all data having been written is made and the routine goes on

as appropriate. When all data has been written, a zero is output as a stop byte, and the disk routine returns to the calling program after the registers have been restored.

Note that the timing for every other byte of data is software timing as ENWD is checked only on even data bytes. My first attempt at this routine was a simple loop which checked ENWD for every byte and, as I said earlier, that results in a loop that is too long. Somehow this algorithm manages to work. But be careful. You cannot execute this algorithm from a slow PROM, such as a 1702 (which requires one or two wait states per reference). You will have to transfer the program to RAM before execution if you wish to store it in slow PROM (Mits does this with their bootstrap loader for the disk, I am told). If you use 2708 PROMs then there is no problem as long as you remember to put the status table and the other storage requirements (a total of ten bytes) in RAM.

**Reading Data**

The READ routine (Fig. 8) works essentially the same as the WRITE routine with a few minor changes. It reads the sync byte and ignores it. Then it reads the first byte of data as a special case to allow setting up a loop similar to that used to write data. When all data has been read it simply returns, as it isn't necessary to read the stop byte.

**Support Routines**

The subroutines MOVE, IN and OUT support the above routines. MOVE simply tests the status of the disk drive until a head move is allowed, and then returns. It prevents the program from trying to step the head in and out faster than it can respond.

IN and OUT are routines to step the head in and out and should be easy to understand from the program listing.

```
0115  7B              MOV A,E ;GET DATA INTO ACCUM FAST
0116  D3  0A          OUT DATA ;AND WWRITE IT TO DISK
0118  5E              MOV E,M ;GET FIRST BYTE OF DATA FROM BUFFER
0119  DB  08         WDAT IN DSTAT ;GET DISK STATUS
011B  1F              RAR ;SHIFT INTO CARRY
011C  DA  19  01      JC WDAT ;LOOP TILL READY
011F  7B              MOV A,E ;GET DATA INTO ACCUM
0120  D3  0A          OUT DATA ;WRITE DATA TO DISK
0122  23              INX H ;POINT TO NEXT BYTE IN BUFFER
0123  7E              MOV A,M ;GET NEXT BYTE READY TO WRITE
0124  15              DCR D ;DECREMENT BYTE COUNTER
0125  CA  30  01      JZ STOP ;IF DONE, GO WRITE STOP BYTE
0128  23              INX H ;POINT TO NEXT BYTE IN BUFFER
0129  5E              MOV E,M ;GET DATA IN REG E SO IT IS HANDY
012A  15              DCR D ;DECREMENT BYTE COUNT
012B  D3  0A          OUT DATA ;TIME TO WRITE THIS BYTE OF DATA
012D  C2  19  01      JNZ WDAT ;IF NOT DONE GO WRITE NEXT BYTE
0130  DB  08         STOP IN DSTAT ;GET DISK STATUS
0132  1F              RAR ;SHIFT INTO CARRY
0133  DA  30  01      JC STOP ;LOOP TILL READY
0136  AF              XRA A ;GET A ZERO IN ACCUM
0137  D3  0A          OUT DATA ;WRITE STOP BYTE
0139  C3  FF  00      JMP EXIT ;LEAVE DISK HANDLER
013C                 ;
013C                 ;TRAP TO BOMB SYSTEM IF RECORD NUMBER IS TOO HIGH —
013C                 ;THIS COULD BE A JUMP BACK TO YOUR SYSTEM TO PRINT AN
013C                 ;APPROPRIATE ERROR MESSAGE IF YOU HAVE THAT CAPABILITY
013C                 ;
013C  C3  3C  01     ERR1 JMP ERR1 ;CLUTSY CALLING PROGRAM HAS BLOWN IT
013F                 ;
013F                 ;
013F                 ;STORAGE REQUIRED BY DISK HANDLER
013F                 ;MAY BE ORG'D TO ANYWHERE YOU HAVE MEMORY IF
013F                 ;YOU WANT THE DISK PROGRAM IN PROM
013F                 ;
013F  00             DRVNM DB 0 ;CONTAINS CURRENT DRIVE NUMBER
0140  80             TRKNM DB 80H ;TABLE OF STATUS BYTES FOR DRIVES 0-3
0141  80              DB 80H ;STATUS BYTES ARE INITIALY SET TO 80 HEX
0142  80              DB 80H ;SO ENABLE ROUTINE WILL INITIALIZE DISK
0143  80              DB 80H ;DRIVES THE FIRST TIME THEY ARE USED
0144                 ;AFTER INITIALIZATION THE STATUS BYTE HOLDS THE
0144                 ;CURRENT TRACK POSITION FOR ITS DRIVE
0144  80             HCS DB 80H ;STATUS FOR HEAD CURRENT SWITCH
0145  00             BYTES DB 0 ;TEMPORARY SAVE SPACE FOR NUMBER OF BYTES
0146                 ;          TO BE INPUT OR OUTPUT
0146  00  00         BUFAD DW 0 ;TEMPORARY SAVE SPACE FOR BUFFER ADDRESS
0148  FF             TEMP DB 0FFH
0149                 ;
0149                 ;
0149                 ;EQUATES
0149                 ;
0149                 DSTAT EQU 8 ;DISK STATUS PORT
0149                 DCONT EQU 9 ;DISK CONTROL PORT
0149                 POS EQU 9 ;SECTOR POSITION PORT
0149                 DATA EQU 10 ;DATA PORT
0149                 HDBIT EQU 4 ;HEAD CONTROL AND TEST BIT
0149                 ENBIT EQU 8 ;DISK ENABLED TEST BIT
0149                 TOBIT EQU 40H ;TRACK 0 TEST BIT
0149                 INBIT EQU 1 ;STEP HEAD IN COMMAND
0149                 OUTBT EQU 2 ;STEP HEAD OUT COMMAND
0149                 MVBIT EQU 2 ;HEAD MOVE TEST BIT
0149                 HCSON EQU 0C0H ;HEAD CURRENT SWITCH AND WRITE ENABLE
0149                 HCOFF EQU 80H ;WRITE ENABLE WITH HEAD CURRENT SWITCH OFF
0149                 SECBT EQU 1 ;START OF SECTOR TEST BIT
0149                 SYNC EQU 0FFH ;SYNC BYTE
0149                 PSW EQU 6 ;CORRECTS DEFICIENCY IN ASSEMBLER
0149                 ;
0149                 ;
0149                 ;
0149                 ;PROGRAM TO TEST DISK I/O ROUTINE
0149                 ;
0149                 ;ASSEMBLE THIS SECTION OF CODE WITH THE DISK ROUTINE
0149                 ;ONLY IF YOU NEEED IT FOR TESTING
0149                 ;
0149                 ;ENTER THE TEST ROUTINE BY BEGINNING EXECUTION AT
0149                 ;EITHER TSTRD OR TSTWT — BE SURE THE STACK IS SET
0149                 ;UPON COMPLETION THE TEST PROGRAM JUMPS TO THE
0149                 ;OPERATING SYSTEM — YOU MUST EQUATE SYS TO BE EQUAL
0149                 ;TO THE APPROPRIATE ADDRESS IN YOUR SYSTEM
0149                 ;IF YOU HAVE NO SYSTEM THEN USE AN ENDLESS LOOP TO
0149                 ;TERMINATE EXECUTION
0149                 ;
0149                 ;TSTAD SHOULD BE EQUATED TO THE ADDRESS OF A 128
0149                 ;BYTE SPACE IN RAM WHICH WILL BE USED FOR I/O BY
0149                 ;THIS TEST PROGRAM
0149                 ;
0149  CD  5F  01     TSTWT CALL SETUP ;SET UP CALLING REGISTERS
014C  CD  05  01      CALL WRITE ;WRITE TO DISK
014F  C3  58  01      JMP EXIT1
0152  CD  5F  01     TSTRD CALL SETUP ;SET UP CALLING REGISTERS
0155  CD  D5  00      CALL READ ;READ FROM DISK
0158  3E  08         EXIT1 MVI A,8 ;UNLOAD HEAD
015A  D3  09          OUT DCONT
015C  C3  0C  10      JMP SYS ;GO TO SYSTEM
015F  21  00  F9     SET UP LXI H,TSTAD ;POINT HL AT BUFFER
0162  11  00  04      LXI D,400H ;LOAD BLOCK NUMBER INTO DE
0165                 ;THIS IS AN ARBITRARY VALUE OF YOUR CHOICE
0165  3E  80          MVI A,128 ;READ OR WRITE 128 BYTES
0167  06  00          MVI B,0 ;USE DRIVE ZERO
0169  C9              RET
016A                 ;
016A                 TSTAD EQU 0F900H ;CONVENIENT RAM FOR TEST BUFFER
016A                 SYS EQU 100CH ;ADDRESS OF MY SYSTEM
```

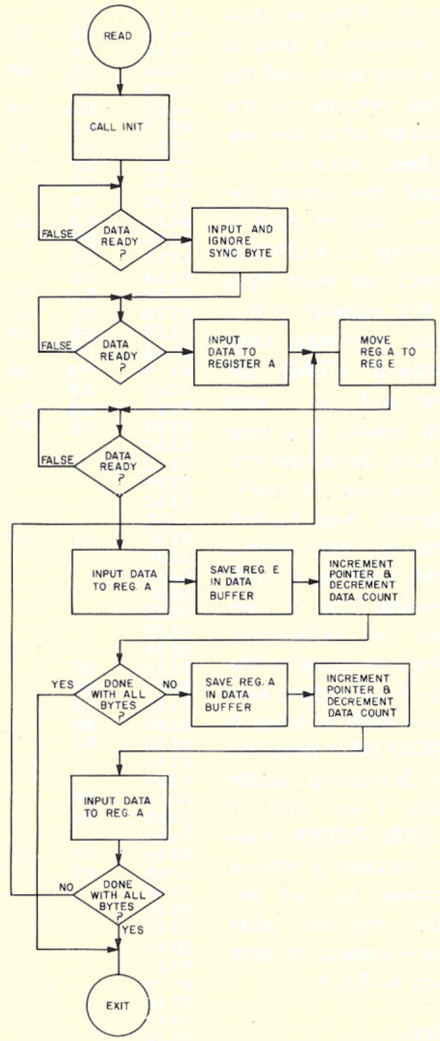



*Fig. 8. Flowchart of the READ routine.*

### Testing

I have included a simple test program which you can use to check the I/O routines. There is not much to it. Simply set up the registers as required and call the READ or WRITE routine. I have included unloading of the head in the test routine. Be sure that you have the stack pointer set up as the I/O routine makes extensive use of the stack.

### Using Disk I/O

This could be a long paragraph if I tried to tell you how to use your disk. So I will describe how I bring up my system as a compromise.

I have this I/O routine ORG'd up above F000 where I have a 2708 PROM. When I turn on my system I begin execution at F000. Some other routines initialize the restart locations in low memory and also the status bytes used by the disk routine. Then a simple program in the PROM brings my system in from disk by reading the first 64 blocks and storing them in memory (using this I/O routine). It unloads the head and branches to the system start address which queries the operator for the correct time and date. Though I read an entire 8K from the disk, my system is somewhat smaller. It was just a convenient choice which allows for changes in the system without changing the PROM. Loading the 8K takes less than a second.

Once the system is loaded, disk I/O goes under system control although the same routine stored in PROM is used for I/O. ■

*Carl Denver Warren II*
*2980 West 235th Apt 12*
*Torrance CA 90505*

# The Axiom EX-800

## *a versatile printer*

When I ventured into the world of personal computing I established certain priorities, the first being to learn what the box could do, and the second to design useful work programs. However, to achieve this goal, some sort of hard-copy device was needed. I quickly surveyed the field and found that most medium priced printers, \$200-\$400, either printed only 40 columns or were restricted to 110 or 300 baud, none of which was acceptable. Consequently, I settled upon the Axiom EX-800, which met all my needs: 1. Variable column size, 2. Selectable baud rate, 3. Easy to interface.

The Axiom EX-800 electrosensitive printer is manufactured by Axiom Corporation of Glendale CA. The printer is not a kit but a professional-quality, fully-assembled machine selling in the \$600-\$700 range, depending upon the interface option selected.

The Axiom is designed as a 120 line-per-minute printer, which in its basic configuration interfaces directly with a parallel input port. However, an optional serial board, which allows for interfacing to RS232C data sources, can be purchased. The printer is small (4 x 9½ x 11 inches), quiet and easy to maintain.

### Electrosensitive Printing Method

The Axiom uses an 8-pin wire printhead that burns off the top coating of aluminized conductive paper. The printhead is in direct contact with the paper at all times and produces 5 x 7 dot matrix characters by selectively supplying current to the print wires.

Photo 2 shows the drive mechanism for the Axiom. The entire drive cam and helical shaft are made of high quality machined metal — not

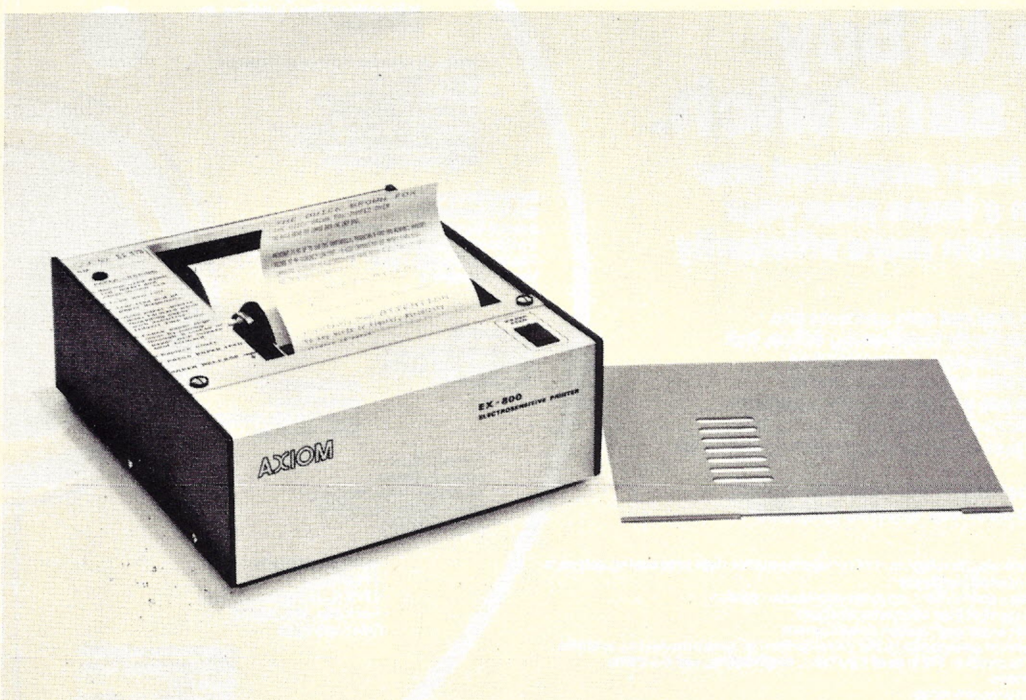



*Photo 1. The Axiom EX-800 electrosensitive printer.*

plastic — parts, which insures reliability. The electronics of the printer is controlled by a microprocessor system that interprets the input codes it receives. Table 1 lists the codes the printer will accept. When a complete line has been stored or a print command received, the print cycle is initiated as follows: the cam in engaged; the helical drive moves the printhead across the paper; the microprocessor sends out the impulses needed to print characters.

**Data Acceptance and Printing Speed**

The reverse channel acknowledge signal from a parallel interface (PIA) or receiver overrun signals from a serial interface (UART) are used to limit the data source inputs to acceptable rates, allowing the system to be used to its greatest capability. data source, the printer will take one second to convert either message to hard copy. Therefore, in order to smooth out the data acceptance rate, a ring buffer is employed. The ring buffer stores the control commands and characters to smooth out the printing rate, but cannot overcome the long term limit of two lines per second printing speed.

**Characters Per Line Options**

The number of characters per line is selectable — 20, 40, and 80. Options may be selected by inserting or removing jumpers on the circuit board. The user manual provides a chart to show each jumper point. An option for mixed character size also can be chosen which allows for software control of the mixed character size. Table 2 contains the software control for various character sizes. For

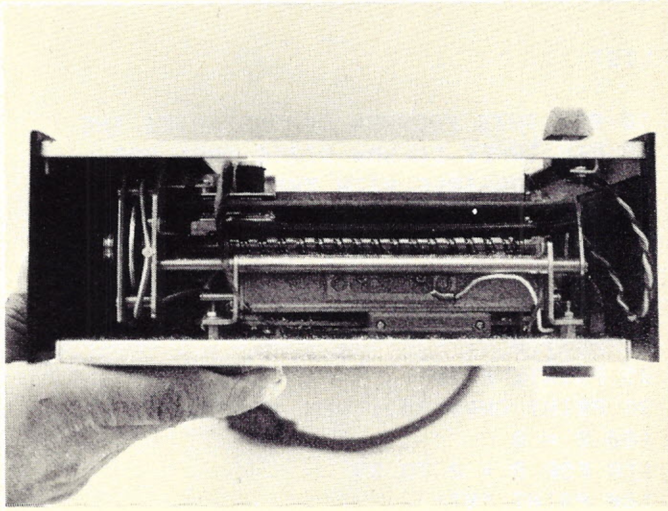


*Photo 2. An internal view of the Axiom printer.*

Axiom is the paper. It is an aluminized conductive paper 5½ inches wide and 250 feet long that sells for $3 per roll. The manufacturer, Nicolet Paper Company of Depere, Wisconsin, sends an evaluation box of four rolls when you purchase the printer.

| LEAST SIGNIFICANT BITS | MOST SIGNIFICANT BITS 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | | p | 0 |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q | 1 |
| 2 | STX | DC2 | " | 2 | B | R | b | r | 2 |
| 3 | ETX | DC3 | # | 3 | C | S | c | s | 3 |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t | 4 |
| 5 | ENQ | NAK | % | 5 | E | U | e | u | 5 |
| 6 | ACK | SYN | & | 6 | F | V | f | v | 6 |
| 7 | BEL | ETB | ' | 7 | G | W | g | w | 7 |
| 8 | BS | CAN | ( | 8 | H | X | h | x | 8 |
| 9 | HT | EM | ) | 9 | I | Y | i | y | 9 |
| A | LF | SUB | * | : | J | Z | j | z | A |
| B | VT | ESC | + | ; | K | [ | k | { | B |
| C | FF | FS | , | < | L | \ | l | : | C |
| D | CR | GS | - | = | M | ] | m | } | D |
| E | SO | RS | . | > | N | ^ | n | ¬ | E |
| F | SI | US | / | ? | O | — | o | DEL | F |

*Table 1. Acceptable input codes.*

This is done by rapidly filling the buffer to insure that a full line is available before the next print cycle is begun. In other words, buffer locations are filled in relationship to the rate the buffer is emptied by printing characters or performing control functions.

The Axiom needs approximately one half second to print a line or perform a linefeed regardless of the number of characters printed. Consequently, this means that for every two linefeeds, or 160 alphanumeric characters transmitted from the formatting outputs, a line may be thought of as having 80 spaces, in which a 20-column character takes up four spaces, a 40-column character two spaces, and an 80-column character one space.

**Paper Characteristics**

The only drawback of the

The paper comes in several grades, but after much experimenting I discovered that stock number 219 appears to be the best of the choices presently available. This grade is lightweight and less susceptible to fingerprinting. However, according to the June issue of *Mini-micro Systems*, Nicolet is planning a white-finish stock in addition to the present dull-aluminum stock.

Due to its highly reflective nature, the paper is extremely easy to copy, and produces excellent results. Program A is an example of how well copying does work since it was necessary to produce a Xerox copy of this program for publication.

**Documentation and Warranty**

Axiom Corporation provides not only a reliable printer, but a complete 44-page operation manual consisting of sections labeled Introduction, Operator checklist, Maintenance, Interface details and signal connections, and Specifications. Appendices cover character codes and schematics. A detailed list and diagram of all mechanical parts are also included. The only deficiency is the lack of a comprehensive troubleshooting guide. Technical assistance offered by Axiom personnel more than makes up for this.

Axiom gives a 90-day all parts and labor guarantee on the EX-800. They feel that if anything is going to happen it will happen during that shakedown period. Also,

| function | ASCII | binary | decimal | octal | hex | keyboard |
|---|---|---|---|---|---|---|
| CHG to 80 | GS | 0011101 | 29 | 035 | 1D | ctrl top bracket |
| CHG to 40 | US | 0011110 | 30 | 036 | 1E | ctrl top arrow |
| CHG to 20 | RS | 0011111 | 31 | 037 | 1F | ctrl back space |

*Table 2. Character size commands.*

```
LIST

10 REM THIS PROGRAM DEMONSTRATES THE
20 REM THREE COLUMN SIZES THAT ARE
30 REM POSSIBLE WITH THE AXIOM EX-800
40 PRINT CHR$(29):REM 80 COLUMNS
50 A=0
60 FOR A = 0 TO 79
70 PRINT "A";
80 NEXT A
85 PRINT: PRINT
90 PRINT CHR$(30): REM 40 COLUMNS
100 B = 0
110 FOR B = 0 TO 39
120 PRINT "B";
130 NEXT B
135 PRINT: PRINT
140 PRINT CHR$(31): REM 20 COLUMNS
150 C = 0
160 FOR C = 0 TO 19
170 PRINT "C";
180 NEXT C
185 PRINT:PRINT
190 END
OK

RUN

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

CCCCCCCCCCCCCCCCCCCC

OK
```

*Program A. Character size demonstration program.*

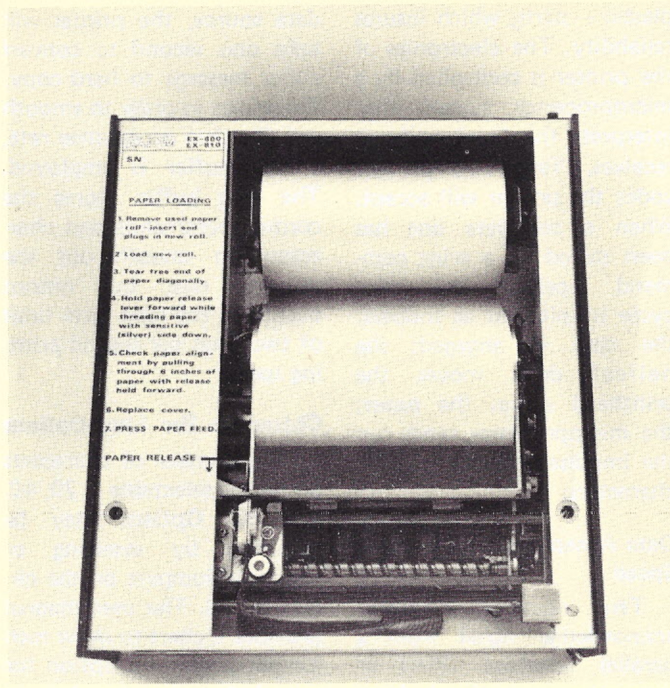



*Photo 3. A top view of the Axiom, loaded with its aluminized paper.*

before the unit is shipped it is subjected to strenuous mechanical and electrical testing. The printed circuit board is run through a one-week burn in when it is subjected to temperature and power changes. This allows Axiom to supply a reliable, guaranteed unit. Even after the warranty period, Axiom provides fast, efficient service. This warranty is the best I have seen for this type of equipment. Of even greater importance — the manufacturer lives up to the warranty.

**Interface Options**

The EX-800 is designed with the user in mind. The printer can be used either in its basic configuration — parallel operation or with a serial adapter, RS232C. All connections for either option are on the back-panel D connector, providing for immediate use of either option by the user.

The optional serial interface, RS232C, is added by plugging in the circuit board, designated by Axiom as S-800A. When the board is in, the back-panel connector on the printer has industry-standard pin designations for RS232C. The S-800A board is designed with a baud rate, parity, and stop-bit selector switch.

**Interfacing the EX-800 to the Mits 680b**

When I originally purchased the Axiom printer, my Mits 680 was configured for communicating through the main board ACIA. Therefore, I used the optional serial interface S-800A. Even though return signal connections are available on the Axiom, I used only two connections: pin 3, serial data in and pin 7, circuit ground. This meant that all data presented to the ACIA would be printed on the Axiom. Consequently, the on-off switch on the printer became my main print control. Also, to prevent receiver overrun and buffer overload, it was necessary for me to add 20 nulls at the end of each BASIC program statement when sending output to the printer.

Using a parallel interface adaptor (PIA) with the 680 reduces the printer's cost, provides full control of the printer and frees the main board ACIA. And the nulls are no longer needed in the program statements. I use the Mits universal input/output board, UIO, which consists of two PIAs and an ACIA.

The Mits UIO provides for

**UIO-to-Axiom Pin Connections**

| Axiom Pin # | UIO IC Pin # | Function | Cable Color |
|---|---|---|---|
| 7 | 5 | GND | Brown |
| 23 | 9 | BIT 7 | Black |
| 14 | 11 | ACK | Red |
| 10 | 12 | STB | Violet |
| 21 | 13 | BIT 6 | Blue |
| 19 | 14 | BIT 5 | Red |
| 18 | 15 | BIT 4 | Black — closet to pin 12 |
| 17 | 16 | BIT 3 | Gray |
| 16 | 17 | BIT 2 | Blue — closet to pin 12 |
| 15 | 18 | BIT 1 | Yellow |

*Table 3. Altair 680b UIO board to Axiom interface.*

address selection of the PIAs by using dip switches. For my purposes I use the lowest setting and set PIAB for FOOE for data and FOOF for control. Using the Mits UIO with the Axiom does present a minor problem. Mits supplies sealed flat cables with incorrect pin connections for the printer, so I rewired the board side connector of the PIA for compatibility. Table 3 shows the correct connections for the UIO IC connector when used with the EX-800.

**Software**

The software for the 680b system is designed to communicate through the main board serial I/O port. Consequently, some software must be changed and an output routine written to handle the interface to the Axiom. Randy Huddleston, in April *Computer Notes*, discussed a couple of ways to handle I/O through the Universal I/O board. However, when only one PIA is being used for a dedicated purpose, a short routine can be used to handle the data transfer. Program B is the assembly listing of an output routine, written by Pat McMullen for use with the Axiom. This routine uses both output addresses found in the ACIA and Baudot versions of the Mits 680b PROM monitor, and was assembled using the Mits 680b assembler/editor.

**Software Patches**

It is necessary to patch existing software (BASIC, editor, assembler/editor) to communicate through the universal I/O board. The patches shown are specifically designed for use with the Axiom, and the output routine shown in Program B. The changes are made after the output routine is loaded, and are made using the monitors M and N commands.

BASIC:

.M 08AE XX 43 – high order byte of output address.

.N 08AF XX AO – low order byte of output address.

```
                            NAM     OUTPUT
                      * WRITTEN BY PAT MC MULLEN
                      * FOR USE WITH THE
                      * MITS 680 UIO.
                      * OUTDIV USE $00F4 EXTENDED
                      * CHARACTER FLAG OF BAUDOT
                      * VERSION OF PROM MONITOR
                      *
                            OPT     S,NOG,PAGE
                      * OUTPUT DEVICE AXIOM PTR
           00F4       OUTDIV EQU    $00F4
           F00E       PIABD  EQU    $F00E    OR $F006 FOR PM-1
           F00F       PIABC  EQU    $F00F    OR $F007 FOR PM-1
                      * OUTPUT SUBRT OF ACIA MONITOR
           FF85       OUTC1  EQU    $$FF85
                      *
                      * INITIALIZE THE PIA'S
                      *
4380                        ORG     $4380
                      * LOAD ACCM A WITH 0
4380 86 00            INIT  LDA A   $00
                      * STORE CTNTS OF A IN CTRL OF PIA
4382 B7 F00F                STA A   PIABC
                      * COMPLEMENT A (FF) TELLS
                      * DDR IS NOW DD
4385 43                     COM A
                      * DDR B IS NOW FOR OUTPUT
4386 B7 F00E                STA A   PIABD
                      * RTN TO DATA PORT INIT
                      * HANDSHAKE
4389 86 2C                  LDA A   #$2C
                      * CB2 STROBE CB1 RESTORE
                      * ACTIVE POS TO NEG
438B B7 F00F                STA A   PIABC
                      * LINE FEED TELLS AXIOM WORKING
                      * REMOVES GARBAGE FROM BUFFER
438E 86 0A                  LDA A   #$0A
4390 B7 F00E                STA A   PIABD
4393 3F                     SWI                    SOFTWR INT RTN TO MON
                      *
                      * OUTPUT ROUTINE
                      *
43A0                        ORG     $43A0
                      * PUSH CNTS OF B ONTO STCK
43A0 37               PRINT PSH B
                      * CHARACTER FLAG
43A1 D6 F4                  LDA B   OUTDIV
                      * OUTPUT TO CRT IF MSB NOT SET
43A3 2A 0D                  BPL     CRT
43A5 F6 F00F          PTR   LDA B   PIABC
                      * OUTPUT TO AXIOM IF MSB SET
43A8 2A FB                  BPL     PTR
                      * DUMMY RUTIN TO CLR HANDSK
43AA 7D F00E                TST     PIABD
                      * TAKES THE CNTS OFF STK
43AD 33                     PUL B
                      * STORES THEM IN DDR OF B
43AE F7 F00E                STA B   PIABD
                      * DECREMENT STK BY 1
43B1 34                     DES
                      * CK CHAR FLAG FOR CRT
43B2 D6 F4            CRT   LDA B   OUTDIV
                      * ROTATE LEFT 1 BIT OR BIT 6
43B4 59                     ROL B
43B5 2A 03                  BPL     RET
                      * GO TO SUBRTN OF MONITOR
43B7 7E FF85                JMP     OUTC1
                      * INCREMENT STK CLEAN OFF THE
                      * GARBAGE
43BA 31               RET   INS
43BB 39                     RTS
                            END

OUTDIV 00F4
PIABD  F00E
PIABC  F00F
OUTC1  FF85
INIT   4380
PRINT  43A0
PTR    43A5
CRT    43B2
RET    43BA

TOTAL ERRORS 00000
```

*Program B. Assembly listing of 680b output routine.*

Assembler/Editor:
.M 022E XX 43 – high order byte of output address.
.N 022F XX AO – low order byte of output address.

Editor:
.M 01EE XX 43 – high order byte of output address.
.N 01EF XX AO – low order byte of output address.

**Initialization Procedure and PIA Test**

When using the UIO, the following procedure is necessary in order to establish communication between the PIA and the printer. The procedure shown is for BASIC, but is similar for the assembler/editor or editor. Regardless of which program is resident, memory location OOF4 must contain the status of bit 6 to determine how the output is handled.

Hex 40 sets bit 6, and data is directed to the CRT only; hex 80 sets bits 6 and 7, and data is directed to both the CRT and the printer; hex CO sets bit 7 only, and data is sent to the printer only with CRT echo turned off. Program C is a BASIC listing demonstrating the use of POKE to test the PIA and output routine. The initialization procedure is shown in Table 4.

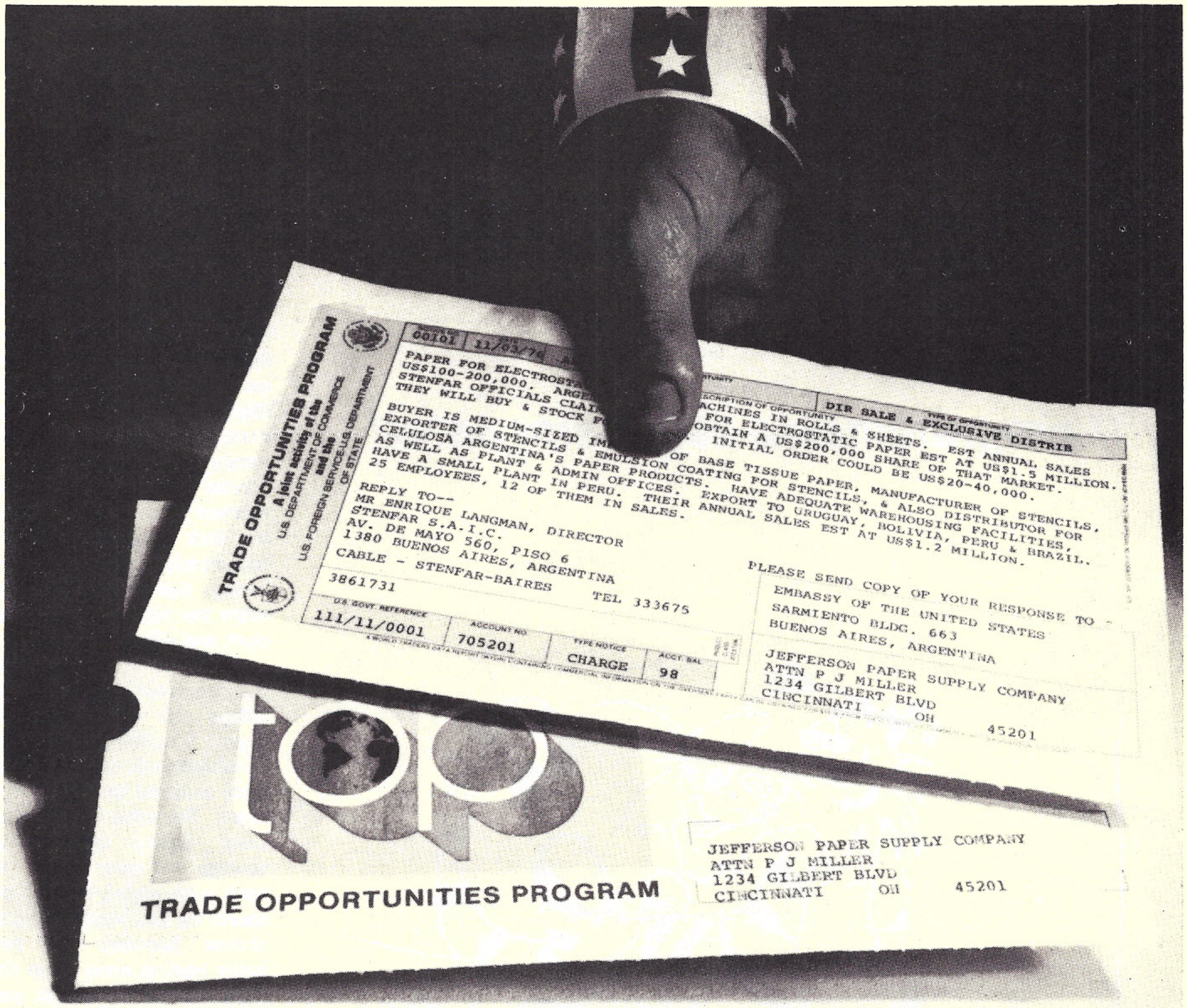
The AXIOM EX-800 electrosensitive printer is a reliable and useful tool. Due to its versatility the printer meets the needs of the hobbyist, software developer and the small businessman. ■

```
10 REM BASIC ROUTINE FOR TESTING
15 REM PIA FOR CRT ONLY, PRINTER
20 REM ONLY AND CRT AND PRINTER
30 REM TEST FOR CRT ONLY
40 POKE 244,64: REM SET BIT 6 CRT
50 PRINT"TEST #1"
60 REM TEST FOR PRINTER ONLY
70 POKE 244,128: REM SET BIT 7
80 PRINT"TEST #2"
85 PRINT:PRINT
90 REM TEST FOR CRT AND PRINTER
91 REM THIS CAUSES OUTPUT TO THE
92 REM PRINTER AND CRT BOTH BIT 6
93 REM AND BIT 7 ARE SET
100 POKE 244.192
110 PRINT"TEST #3"
999 END
```

*Program C. BASIC listing for testing PIA.*

1. Using the monitor's L command load BASIC.
2. Jump to location 0000 ( .J 0000 )
   a. Set memory size to 16583.
   b. Set output size to 80.
   c. Keep or delete arithmetic functions depending on your need.
   d. Return to PROM monitor.
3. Use the monitor's L command, and load the OUTPUT routine.
4. Use the monitor's M and N commands and change the following locations.
   a. .M 00F2 XX FF – causes a return to BASIC on initialization.
   b. .M 00F4 XX 40 – sets bit 6 CRT only.
   c. .M 08AE FF 43 – high order byte of output address.
   d. .N 08AF 81 A0 – low order byte of output address.
5. Use the monitor's J command to jump to the output location 4380 this initializes the PIA, sets up a handshake between the computer and the printer and returns to BASIC.

*Table 4. Initialization procedure.*




# To get overseas trade leads like this, you could open offices in 127 countries.

# Or use our computer.

The U.S. Commerce Department's computer-operated Trade Opportunities Program (TOP) can supply you with immediate, continuing, specific leads tailored to your sales objectives for any of 127 countries. So if you can't be all over the world at one time, we've got the answer. Send us the coupon now and see how you can find where the live leads are living.

Secretary of Commerce
U.S. Department of Commerce, BIC-9A
Washington, D.C. 20230

Please tell me more about the Trade Opportunities Program (TOP).

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City __________ State __________ Zip __________

Ad Council

A Public Service of This Magazine & The Advertising Council

*Mike O' Connell*
*5835-A Brett*
*Fort Knox KY 40121*

# Tiger Trouble!

## *TI programmable-calculator safari*


Your long-awaited camera safari has been a great success. Now, though, your party faces potential tragedy. While traveling down the Ngabiki River, one of the boat's crewmen has fallen seriously ill, and your guide fears he may die without medical attention. Fortunately, a doctor lives in a nearby village, which can be reached from the next landing dock along the river; *un*fortunately, a legendary man-eating tiger is rumored to roam the region between the dock and village.

Someone must undertake the dangerous journey to the village and return with the doctor. You have — naturally — volunteered. Unarmed, you hope to avoid the tiger — if it exists. The other members of your party salute your courage and leave you at the dock with wishes of luck and speed. As you stand trying to choose between the two paths leading away from the dock toward the village, you hear the tiger roar.

Tiger Trouble reverses the traditional game roles, in which the player normally hunts or tracks an elusive prey, through the combined use of random number generation and conditional transfer functions. For more information on the Texas Instruments programmable calculators, their conditional transfer functions and random number generation, see Pete Stark's articles in issues 1 and 2 of *Kilobaud* (Jan. and Feb. 1977). Briefly, the calculator employs a simple algorithm to generate pseudo-random numbers.

In this program, as in Pete Stark's Submarine game (*Kilobaud,* No. 2, p. 70), a seed number between 0 and 1 is multiplied by $7^9$ (40,353,607), and then divided by $10^5$ (100,000).

Digits to the left of the decimal point are discarded, and the remainder is a random number between 0 and 1 (and also the new seed number).

Multiplying the first random number by 21 and rounding to an integer gives an initial location for our tiger; successive random numbers between 0 and 1 are subsequent moves by the tiger; sucessive random numbers between 0 and 1 are multiplied by 4 and then rounded to an integer value of 0, 1, 2 or 3. Here is where the conditional transfer function operates to reverse traditional game roles.

If the tiger's location is numerically greater than the player's location, the integer value generated (0,1,2,3) is subtracted from the tiger's location; otherwise, it is added to the tiger's location. In other words, the tiger will move 1, 2, or 3 spaces closer to the player's position, or will remain in place after each move the player makes. Other conditional transfers are used to end the game if the tiger catches the player, and to prevent the tiger from straying into the Ngabiki River (negative numbers). See Fig. 1.

### Game Rules

The basic operating rule of the game is that the tiger stalks the player. Rules of play and comments follow.

1. A player may take any route to reach the village and return (see the game board configuration, Fig. 2), provided he moves one numbered location at a time along a solid path. A player need not move to locations in numerical sequence and may go forward or backward.

2. In addition to the landing and the village, the game board contains four sectors: the swamp (locations 1-5), forest (locations 6-10), plains (locations 11-15) and hills

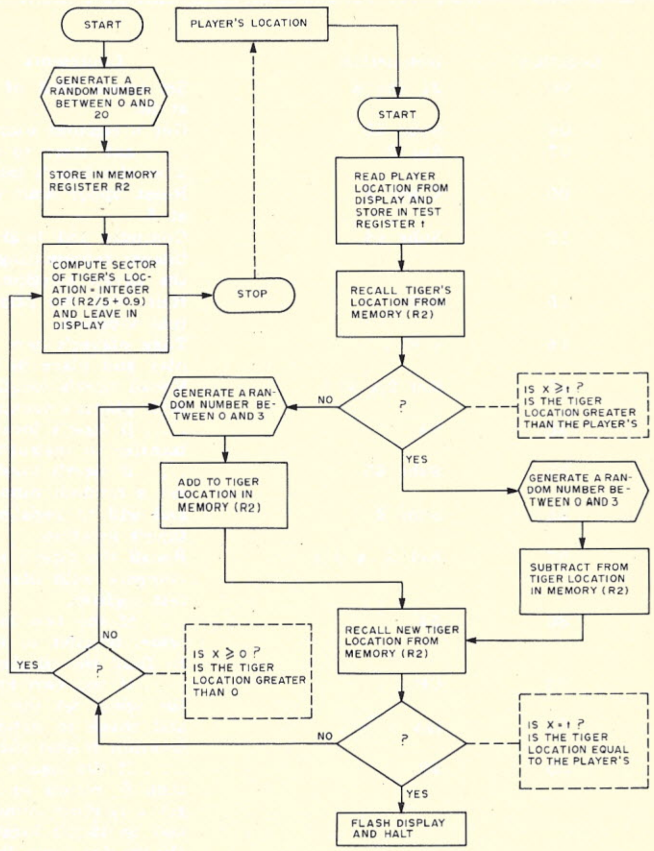



*Fig. 1. Flowchart.*

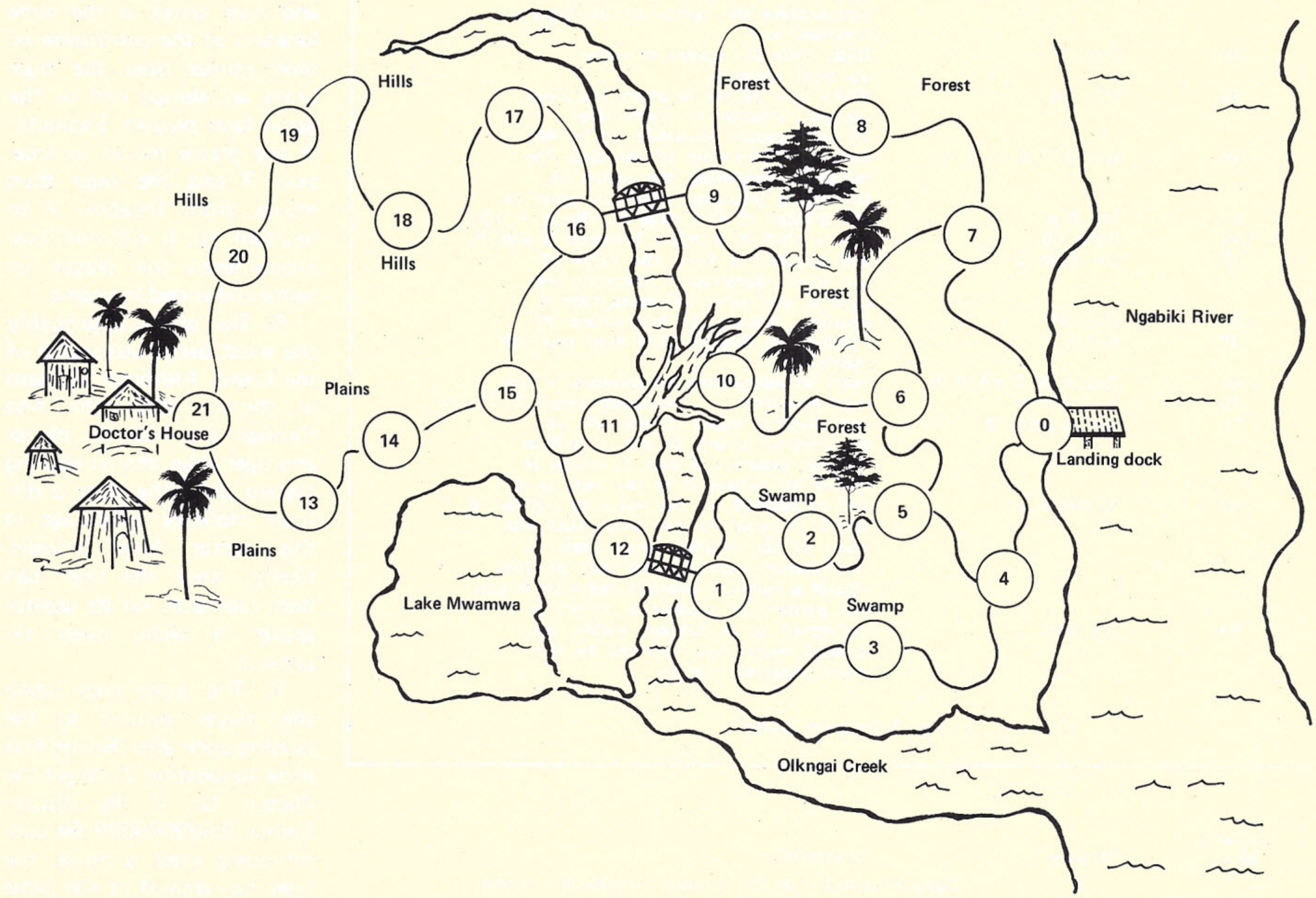



*Fig. 2. Game board.*

| Location | Instruction | Comments |
|---|---|---|
| 00 | 21 Sto 8 | Set upper limit of random number at 20. |
| 04 | Subr 65 | Get a random number between 0 and 20 |
| 07 | Sto 2 | . . . and store in memory register 2 as the tiger's initial location. |
| 09 | 4 Sto 8 | Reset upper limit of random number at 3. |
| 12 | Subr 44 | Compute and hold for display the integer representing the sector of the tiger's location, and |
| 15 | R/S | Halt. Wait for player to enter first location. |
| 16 | $x \gtrless t$ | Take player's new location from display and place in test register t. |
| 17 | Rcl 2,$x \geqslant t$ | Recall tiger's location and compare with player's location in test register, |
| 20 | 54 | . . . if tiger's location is greater, transfer to instruction 54. |
| 22 | Subr 65 | . . . if tiger's location is less, get a random number between 0 and 3, |
| 25 | Sum 2 | and add to register 2 to adjust the tiger's location. |
| 27 | Rcl 2, x = t | Recall the tiger's new location and compare with player's location in test register, |
| 30 | 63 | . . . if the two locations are the same, transfer to instruction 63 to flash the display. |
| 32 | CP | . . . if the two locations are not the same, set the test register to 0, |
| 33 | Inv $x \geqslant t$ | and check to determine if the tiger's location is less than 0, |
| 35 | 22 | . . . if the tiger's location *is* less than 0, return to instruction 22 (to get a random number between 0 and 3, add to tiger's location, and repeat checks for same locations and a negative tiger location). |
| 37 | Subr 44 | . . . if the tiger's location is not less than 0, transfer to instruction 44 for the subroutine to compute and hold for display the integer representing the sector of the tiger's location, and |
| 40 | R/S | Halt. Wait for player to enter his next location. |
| 41 | Gto 16 | When new player location has been entered, transfer to instruction 16 (to execute sequence 16 to 40). |
| 44 | Rcl 2 ÷ 5 + .9 = | Start of subroutine to calculate the integer representing the sector of the tiger's location, which is equal to |
| 52 | Int Rtn | the integer of the value of ($R_{2/5}$ + 0.9). |
| 54 | Subr 65 | Get a random number between 0 and 3, |
| 57 | Inv Sum 2 | and subtract it from the value of the tiger's location in memory register 2, and after adjusting tiger's |
| 60 | Gto 27 | location, transfer to instruction 27. |
| 63 | 0 1/x | Flashes the display (1/0 does not calculate). |
| 65 | Rcl 9 X 7 $y^x$ 9 X | Start of subroutine to generate a ran- |
| 72 | 5 ± $10^x$ = | dom number . . . calculates (seed X $7^9$)/$10^5$, |
| 76 | Inv Int Sto 9 | and discards digits to the left of the decimal point, to get a random number between 0 and 1, which is stored in register 9 as the new seed, |
| 80 | X Rcl 8 = | and multiplied by the value in register 8, which is equal to one more than the value of the desired upper limit of the random number, and that product equals a random number between 0 and the number in register 8, which is |
| 84 | Int Rtn | converted to an integer within the desired range, and returned to the main program routine. |

*Fig. 3. Program listing.*

| Player's Move | Display | Comments |
|---|---|---|
| - - | 1 | Tiger is initially in the swamp — take the forest path. |
| 7 | 1 | Tiger is still in the swamp — continue toward |

(locations 16-20). After each set of moves by player and tiger, the tiger will roar, partially betraying its location. This roar is "heard" as a displayed integer, indicating the sector in which the tiger is located. (A display of 1 means the tiger is in the swamp; 2 means forest; 3 means plains; 4 means hills. The tiger may go to the landing dock, and the display will be 0, or enter the village, causing a 5 to be shown.)

3. Both player and tiger cross Olkngai Creek. The player crosses at the bridge connecting locations 1 and 12, or at the one between locations 9 and 16; because these are man-made and appear strange, the tiger will not cross them. The tiger will cross only at the fallen tree between locations 10 and 11. The player cannot cross this.

4. Unlike the player, the tiger does not travel on the paths; instead, it travels freely through the area to arrive at numbered locations on a path. And only when player and tiger arrive at the same location *at the conclusion of their moves* does the tiger bring an abrupt end to the game (and player). Example: If the player moves to location 9 and the tiger then moves from location 7 to location 10, it will *not* have encountered the player en route and ended the game.

5. The plains is probably the most dangerous sector of the board. Although the path is shorter than the one through the hills for player and tiger alike, the numbering system gives the tiger a distinct mobility advantage in this sector (not unrealistically, since the tiger can best capitalize on its greater speed in wide, open expanses).

6. The game ends when the player returns to the landing dock after having first gone to location 21 to get the doctor. Or, if the display flashes 9.999999999 99 continuously after a move, the tiger has arrived at the same location as the player, and,

well . . . sorry, but the game ends that way, too.

It appears simple, but the odds (and the program) are with the tiger, and it is difficult to win. To play, note the rules and comments above, enter the program as shown in Fig. 3 (remember to put a seed number in memory register 9) and press the R/S button. You are now at the dock, and the number displayed represents the sector of the tiger's initial location. To move, enter the number of the location to which you wish to advance and press R/S. That's all there is to it.

Good luck and safe journey! ■

| | | |
|---|---|---|
| | | the bridge at location 9. |
| 8 | 2 | Tiger is pursuing and is in the forest. |
| 9 | 2 | At the bridge — tiger also still in forest. |
| 16 | 2 | Tiger in forest — take mountain (hills) path. |
| 17 | 2 | Tiger still in forest — must be lost — continue. |
| 18 | 3 | Tiger pursuing — crossed log into plains area. |
| 19 | 3 | Tiger searching plains — continue on hills path. |
| 20 | 4 | Tiger has entered hills in pursuit. |
| 21 | 4 | Reached doctor — must return to dock — since tiger in hills, take plains path. |
| 13 | 3 | Oops — tiger returned to plains — back to village to avoid. |
| 21 | 4 | Tiger has returned to hills — has us well boxed in — will risk the plains path. |
| 13 | 3 | Tiger is back in plains — proceed with caution. |
| 14 | 3 | Tiger is searching plains — will risk continuing. |
| 15 | 3 | Tiger still searching plains — head for bridge at location 12. |
| 12 | 3 | At the bridge — tiger still searching plains — cross into swamp. |
| 1 | 3 | Across bridge — tiger still on plains — should be safe — continue to dock. |
| 2 | 2 | Oh-oh — tiger has crossed log into forest in pursuit — head for dock. |
| 5 | 2 | Tiger moving through forest — race for dock. |
| 4 | 2 | Tiger still in forest — just one more move to landing dock. |
| 0 | 2 | Reached dock safely — tiger never made it through the forest — score one for us. |

*Fig. 4. Sample run (with seed = 0.9). Note: Entire sequence should be duplicated with the same initial seed and identical moves, and can be used as a program check.*

*John P. Bauernschub, Jr.*
*14809 Clavel St.*
*Rockville MD 20853*

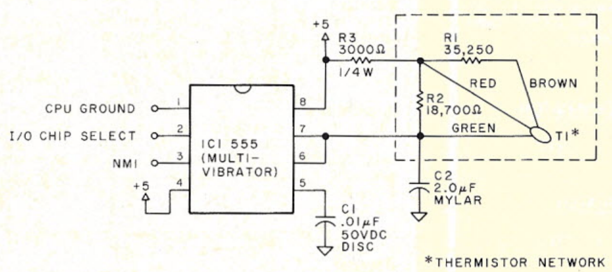
# Temperature Sensing

## *some exciting possibilities!*

Utilize your computer: Let it control the temperature in your home. The computer can be much more flexible than an ordinary thermostat because it can schedule the different temperature settings for personal comfort and energy savings. It can also employ multiple temperature sensors to maintain the comfort level where and when you want it — such as in the bedroom at night, downstairs during the day and in the family room in the evening.

The purpose of this article is to show how a simple and accurate temperature sensor can be made and how to interface and control it with a computer. The software for controlling, calibrating and running the thermometer is included and explained. A secondary purpose is to show how to write an assembler program that can be relocated anywhere in memory without changes. A program that will reposition it in memory is included.

This is written for the SWTP 6800 owner, but the interface and software are readily adaptable to other applications that are dependent on a variable resistance, such as joysticks. The techniques can also be applied to other computers.




*2.0 uF mylar capacitor, 50 WVDC
Allied Electronics catalog no. 852-1411
Type EWF0520 $1.80

**Thermistor network YSI 44203
Allied Electronics catalog no. 997R3002
$14.53 includes R1 and R2

*Fig. 1. Thermometer schematic.*

### Hardware

The temperature sensor electronics described here are sufficiently sensitive to measure one-tenth of a degree Fahrenheit. Only four connections to the SWTP 6800 computer are required — two for signals and two more for power.

Fig. 1 is the electrical schematic of the temperature-measurement device. This is a common technique of using a 555 as a monostable multivibrator. The temperature sensor is a thermilinear thermistor network that has a linear resistance change of -127.096 Ohms per degree Celsius over the range of -30 to +50°C. R1 and R2 come with the thermistor as separate components. R3 is used to limit the current from the 5 volt source. The 2.0 uF capacitor should be a quality mylar capacitor. The thermistor network and the 2.0 uF capacitor were purchased from Allied Electronics. The other components can be acquired from many sources.

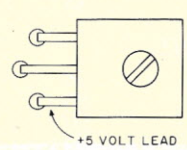
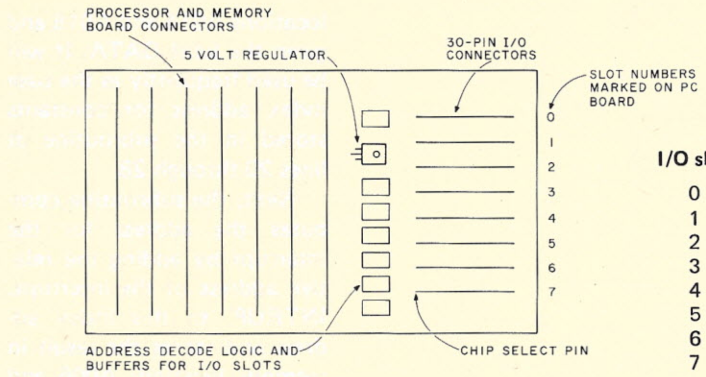



*Fig. 2. Five volt regulator IC on SWTP 6800 motherboard.*

The device is turned on by a decoded address pulse that switches the input on pin 2 of IC1 from logic 1 to 0. The output on pin 3 goes from logic 0 to 1, and the R/C timing circuit charges the 2.0 uF capacitor. When the capacitor reaches 2/3 the input voltage, the device is turned off and the output on pin 3 goes to logic 0. The output pin is connected to the non-maskable interrupt (NMI) line of the 6800. This will interrupt a counting loop in the software and cause a jump to the processing routine.

This circuit requires four connections to the SWTP motherboard. Ground and




| I/O slot | Address | |
|---|---|---|
| 0 | 8000 | |
| 1 | 8004 | Control |
| 2 | 8008 | |
| 3 | 800C | |
| 4 | 8010 | |
| 5 | 8014 | |
| 6 | 8018 | |
| 7 | 801C | Printer |

*Fig. 3. SWTP 6800 motherboard layout.*

the NMI lines are marked on the motherboard for both the memory connectors and the I/O port interfaces. Connectors can be purchased from SWTP to mate with these connectors, or wires can be soldered to the pins. The +5 volts can be obtained simply by soldering a wire to the regulator on the motherboard. See Fig. 2.

The address decoding is performed on the motherboard and is available on the end pin of each I/O interface connector. It is unmarked but is next to the RESET pin. Fig. 3 shows where these connectors and pins are located. Slots #1 and #7 are reserved for the control terminal and the printer. The addresses for the I/O slots are given in Table 1. The subroutine uses the address of slot #6. If you use a different slot, its address must be entered in the subroutine at line 23, bytes 6 and 7.

This method is a simple way of interfacing the temperature-sensing electronics to the computer. A more elegant way would be to use a parallel interface board. A word of caution: Always have the power off when making any connections to the CPU or temperature-sensing circuit. Do not allow the conductors of the wires you are bringing out of the CPU to come in contact with any other wires, printed circuits or components.

**Software**

Four programs are provided in this article. The first is a general purpose subroutine that handles the temperature-sensing and display functions. It has been written so that it can be moved anywhere in memory without changing one byte in it. Some techniques required for this feature will be described. The second program sets the mode byte for the subroutine, calls the subroutine, then returns to MIKBUG control. It can be repeatedly executed to display the temperature by pressing G. The third program is a utility to relocate the temperature-sensing subroutine to where you want it. These programs can also be placed anywhere in memory without any changes. The fourth is a BASIC program. This allows the use of the temperature in your BASIC program.

The control of the temperature sensor and the scaling of the measurement are done in a subroutine that is flexible and adaptable to many applications. You can place it anywhere in memory without change. For this reason, when it is called it must perform some housekeeping. First it must locate itself. The BSR instruction causes the address of MULT to be pushed into the stack. This address is then pulled from the stack at FIND and stored in the scratchpad memory at

*Program 1. Temperature sensing subroutine.*

```
00001                        NAM   THERMOMETER
00002                 *      July 27, 1977
00003                        OPT   O
00004       A014      DATA   EQU   $A014
00005       A016      SAVSTK EQU   $A016
00006       A018      BINTMP EQU   $A018
00007       A01A      COVDTA EQU   $A01A
00008       A01C      OUTSTR EQU   $A01C
00009       A01E      SAVEA  EQU   $A01E
00010       A01F      SAVEX  EQU   $A01F
00011       A021      SAVEX1 EQU   $A021
00012       A023      MODE   EQU   $A023
00013       E07E      PDATA1 EQU   $E07E
00014       E0E3      CONTRL EQU   $E0E3
00015                 *
00016                 *
00017                 *
00018 0000                   ORG   $0000
00019 0000  8D 1E     BEGSUB BSR   FIND
00020 0002  18        MULT   FCB   24      MULTIPLICATION CONSTANT
00021 0003  2C        DIVIDE FCB   44      DIVISION CONSTANT
00022 0004  120A      INITX  FCB   4618    INITIALIZATION CONSTANT
00023 0006  8018      TURNON FCB   $8018   I/O PORT ADDRESS
00024 0008  2710             FDB   10000
00025 000A  03E8             FDB   1000
```

```
00026  000C  0064                FDB   100
00027  000E  000A                FDB   10
00028  0010  0001                FDB   1
00029  0012  0D0A                FDB   $0D,0A,,,
       0014  0000
       0016  0000
       0018  0000
00030  001A  0400                FDB   $0400,,
       001C  0000
       001E  0000
00031  0020  32       FIND       PUL A
00032  0021  33                  PUL B
00033  0022  B7 A014             STA A  DATA
00034  0025  F7 A015             STA B  DATA+1
00035  0028  CB 69               ADD B  #INTRUP-MULT
00036  002A  89 00               ADC A  #0
00037  002C  B7 A006             STA A $A006
00038  002F  F7 A007             STA B $A007
00039  0032  B6 A014             LDA A DATA
00040  0035  F6 A015             LDA B DATA+1
00041  0038  CB 06               ADD B  #6
00042  003A  89 00               ADC A  #0
00043  003C  B7 A01A             STA A  COVDTA
00044  003F  F7 A01B             STA B  COVDTA+1
00045  0042  CB 0A               ADD B  #10
00046  0044  89 00               ADC A  #0
00047  0046  B7 A01C             STA A  OUTSTR
00048  0049  F7 A01D             STA B  OUTSTR+1
00049  004C  01       CYCLE      NOP
00050  004D  0F                  SEI
00051  004E  BF A016             STS    SAVSTK
00052  0051  FE A014             LDX    DATA
00053  0054  EE 04               LDX    4,X
00054  0056  A7 00               STA A  X           START THERMOMETER CIRCUIT
00055  0058  7D A023             TST    MODE
00056  005B  26 06               BNE    DEGREE
00057                 *                 CALIBRATION COUNTER
00058  005D  CE 0000             LDX    #0
00059  0060  08       LOOP1      INX
00060  0061  20 FD               BRA    LOOP1
00061                 *                 TEMPERATURE COUNTER
00062  0063  FE A014  DEGREE     LDX    DATA
00063  0066  EE 02               LDX    2,X
00064  0068  09       LOOP       DEX
00065  0069  20 FD               BRA    LOOP
```

locations A014 and A015 and given the label DATA. It will be used frequently as the base index address for constants stored in the subroutine at lines 20 through 28.

Next, the subroutine computes the address for the interrupt by adding the relative address of the interrupt, INTRUP, to this index address and stores the result in memory locations A006 and A007. The subroutine makes use of the scratchpad memory from locations A014 through A023. All branching within the subroutine uses relative addressing. In one case, a stepstone branch is used to extend the relative address range (lines 151 to 88 to 74).

The temperature sensor is turned on by any instruction referencing the selected I/O port address. This is done in line 54 of the subroutine by the STA A X instruction. A constant is then loaded into the index register. The subroutine enters a loop that decrements the index register while the capacitor is charging. The loop is stopped by the nonmaskable interrupt caused by the temperature-sensing circuit. The contents of the index register are automatically pushed into the stack by the interrupt. The interrupt address stored in memory locations A006 and A007 is the next step in the subroutine.

After the interrupt, the stack pointer is transferred to the index register, and the results of the counter are retrieved and loaded into registers A and B. This value is then multiplied and divided by constants, converted to decimal and displayed on the terminal. All numbers used in the computation are two-byte binary integers. The calibration constants are such that the answer is ten times the number of degrees Fahrenheit. Subroutine lines 184 through 187 move the tenths portion of the answer over one byte and insert a decimal point. Lines 191 through 198 replace leading zeros with

blanks. The multiplication and division routines are slightly modified versions of those found in the *Motorola 6800 Programming Manual*.

Subroutine line 51 saved the stack pointer before the interrupt and the math routines. In lines 70 and 74 the stack pointer is restored to recover from the stack changes caused by these routines. Therefore, there is no need for a return from the interrupt.

Because this is a general-purpose subroutine, the temperature is saved in three formats for use by your programs. The first is a string of eight bytes in ASCII format that includes a carriage return, line feed, decimal point and spaces in place of leading zeros. It can be found in line 29 of the listing. Its actual starting address is computed and saved in memory location A01C, labeled OUTSTR, for use by your routines. The ninth byte is a string terminator, hex 04.

The other two formats contain an integer value of ten times the temperature. As an example, a decimal value of 793 found would really represent 79.3°F. The second format is binary-coded decimal in five bytes. It is located after the ASCII format string terminator and is relative byte #9 from the address stored in A0IC. 79.3 would be found as 00 00 07 09 03. The third format is unsigned binary of two bytes and is stored in memory locations A018 and A019 under the label BINTMP.

When you are calling the subroutine from your program, you may not want the temperature displayed. This can be deleted by changing the LDX instruction in line 83, subroutine byte 8C, to RTS. This is done by changing FE to 39 in byte 8C.

Space has been reserved in the subroutine for you to jump to your program rather than return from the subroutine. You can replace the RTS in line 85, byte 92, and the following two NOPs with

```
00066                   *           INTERUPT ROUTINE
00067  006B  30         INTRUP TSX
00068  006C  A6 03             LDA A  3,X        RECOVER COUNTER DATA
00069  006E  E6 04             LDA B  4,X
00070  0070  BE A016           LDS    SAVSTK     RESTORE STACK AFTER INTERUPT
                                                                      [sic]
00071  0073  7D A023           TST    MODE
00072  0076  27 02             BEQ    A1
00073  0078  20 1D             BRA    MATH
00074  007A  BE A016 A1        LDS    SAVSTK     RESTORE STACK AFTER MATH
00075  007D  01                NOP
00076  007E  0E                CLI
00077  007F  FE A01C           LDX    OUTSTR
00078  0082  08                INX
00079  0083  08                INX
00080  0084  B7 A018           STA A  BINTMP     SAVE BINARY TEMPERATURE
00081  0087  F7 A019           STA B  BINTMP+1
00082  008A  8D 76             BSR    CVBTD
00083  008C  FE A01C           LDX    OUTSTR
00084  008F  BD E07E           JSR    PDATA1
00085  0092  39                RTS               RETURN TO CONTROL PROGRAM
00086  0093  01                NOP
00087  0094  01                NOP
00088  0095  20 E3   A2        BRA    A1
00089                *                MULTIPLY AND DIVIDE
00090  0097  37      MATH      PSH B
00091  0098  36                PSH A
00092  0099  FE A014           LDX    DATA
00093  009C  A6 00             LDA A  X
00094  009E  36                PSH A
00095  009F  4F                CLR A
00096  00A0  36                PSH A
00097  00A1  86 10             LDA A  #16
00098  00A3  36                PSH A
00099  00A4  30                TSX
00100  00A5  A6 03             LDA A  3,X
00101  00A7  58      M1        ASL B
00102  00A8  49                ROL A
00103  00A9  68 02             ASL    2,X
00104  00AB  69 01             ROL    1,X
00105  00AD  24 04             BCC    M2
00106  00AF  EB 04             ADD B  4,X
00107  00B1  A9 03             ADC A  3,X
00108  00B3  6A 00   M2        DEC    0,X
00109  00B5  26 F0             BNE    M1
00110  00B7  37      D1        PSH B
```

```
00111 00B8 36                  PSH A
00112 00B9 4F                  CLR A
00113 00BA FE A014             LDX     DATA
00114 00BD E6 01               LDA A   1,X
00115 00BF 37                  PSH B
00116 00C0 36                  PSH A
00117 00C1 34                  DES
00118 00C2 30                  TSX
00119 00C3 86 01               LDA A   #1
00120 00C5 6D 01               TST     1,X
00121 00C7 2B 0B               BMI     D3
00122 00C9 4C          D2      INC A
00123 00CA 68 02               ASL     2,X
00124 00CC 69 01               ROL     1,X
00125 00CE 2B 04               BMI     D3
00126 00D0 81 11               CMP A   #17
00127 00D2 26 F5               BNE     D2
00128 00D4 A7 00       D3      STA A   X
00129 00D6 A6 03               LDA A   3,X
00130 00D8 E6 04               LDA B   4,X
00131 00DA 6F 03               CLR     3,X
00132 00DC 6F 04               CLR     4,X
00133 00DE E0 02       D4      SUB B   2,X
00134 00E0 A2 01               SBC A   1,X
00135 00E2 24 07               BCC     D5
00136 00E4 EB 02               ADD B   2,X
00137 00E6 A9 01               ADC A   1,X
00138 00E8 0C                  CLC
00139 00E9 20 01               BRA     D6
00140 00EB 0D          D5      SEC
00141 00EC 69 04       D6      ROL     4,X
00142 00EE 69 03               ROL     3,X
00143 00F0 64 01               LSR     1,X
00144 00F2 66 02               ROR     2,X
00145 00F4 6A 00               DEC     X
00146 00F6 26 E6               BNE     D4
00147 00F8 A6 03               LDA A   3,X
00148 00FA E6 04               LDA B   4,X
00149 00FC CB 00               ADD B   #0
00150 00FE 89 00               ADC A   #0
00151 0100 20 93               BRA     A2
00152                  *               CONVERT BINARY TO DECIMAL
00153 0102 FF A01F CVBTD       STX     SAVEX
00154 0105 FE A01A             LDX     COVDTA
00155 0108 7F A01E C1          CLR     SAVEA
```

your jump instruction. The NOPs were placed there so the relative addresses would not have to be changed if you wanted to use a JMP instead of the RTS.

One final note on the subroutine. When it is assembled, line 35 generates an error 210 on the first pass. This is only a warning because the result of the subtraction of double byte numbers, the addresses of INTRUP and MULT, could possibly exceed the one data byte of the instruction.

The second program is an example of how you would call the subroutine from your program. Before the subroutine is called, the mode switch is set for temperature sensing or calibration. This is explained in the calibration section. When execution of this program begins, the stack is decremented twice to preserve the start address in memory locations A048 and A049. This is a good idea for all 6800 MIKBUG-controlled assembler programs because the start address does not have to be reloaded to restart the program.

The third program is a utility routine for relocating the subroutine. To run this routine, load the address of the first byte of the program to be moved into memory locations A02A and A02B and the address of the last byte into memory locations A02C and A02D. Then load the first address of where you want to move the program into memory locations A02E and A02F. Load this routine's start address, A014, into memory locations A048 and A049 and execute the routine. This third program will be used in preparing the subroutine to be called by BASIC.

For those who prefer to write their programs in BASIC, I have included a short BASIC routine. It calls the temperature-sensing subroutine and recovers the temperature in the BASIC variable T. This is written for SWTP 8K BASIC version 2.0. These instructions will proba-

# Overseas agents and distributors are waiting to sell your products in foreign markets.

## Find out how to find them. Fast.

**You've got a great product. With great domestic sales. And now you want your share of the annual $107 billion U.S. export market. But who's going to handle that product thousands of miles from home? The U.S. Commerce Department can tell you. Because through our Agent/Distributor Service, we'll match your product with up to six firms that are ready, willing and able to represent you in the countries you choose. To learn more, write Secretary of Commerce, BIC 11C, Washington, D.C. 20230.**

Ad Council

A Public Service of This Magazine & The Advertising Council

---

# Computer Corner OF NEW JERSEY

**(201) 835-7080**
**Telex - 130-376**
answer back AMCALL POMT

240 WANAQUE AVE., POMPTON LAKES, N.J. 07442

## micro-mini computer center

**Now in the North Jersey Area!**

**AUTHORIZED DEALER FOR:**

- Southwest Technical Products Corp. • Micro-Term
- Jim-Pak Electronic Components • Kim-1 • Sanyo
- Ohio Scientific • Technico, Inc. • Hayden Books
- PolyMorphic Systems • Seals Electronics, Inc.

## *LOOK AT THESE PRICES!*

**Computer Corner** OF NEW JERSEY ***Offers You These Unbeatable Values!***

You can now buy the New **Netronics ELF II, RCA COSMAC** microprocessor, minicomputer for only **$139.95** wired & tested!

Also. . . **VERBATIM Removable Magnetic Storage Media**

| | 1-9 | 10-25 | 26-100 |
|---|---|---|---|
| **Minidiskettes** | 4.50 | 4.25 | 3.75 |

**MD525-01** (Soft Sector) for: Intelligent Systems, Magnavox, Microkit, and Vector Graphics

**MD525-10** (10 Sector, Hard) for: Digi-log, North Star, Polymorphic, Tei Inc., and Wang

**MD525-16** (16 Sector, Hard) for: Altair, Comtek, Micropolis, R2E, Realistic Controls, and Teleray-Research, Inc.

| | 1-9 | 10-25 | 26-100 |
|---|---|---|---|
| **Standard Size Diskettes** | 5.50 | 5.25 | 4.75 |

**FD34-1000** (Soft Sector, IBM Std.)
**FD32-1000** (Hard Sector, Inner dia.)
**FD65-1000** (Hard Sector, Outer dia.)

**TO ORDER:**
Send check or money order and include $1.00 for shipping, 1.00 (optional) for insurance, and please include 5% sales tax if you are a New Jersey state resident. Phone orders accepted with Mastercharge or Visa.

---

# H8® & TRS-80® SOFTWARE

| ORDER CODE | DESCRIPTION | MINIMUM SYSTEM | PRICE EACH |
|---|---|---|---|
| HEATHKIT® H8 SOFTWARE | | | |
| | Benton Harbor Basic Games | | |
| H8-BG1 | BIORYTHM, CHOMP, TEASER, BATUM CRAPS, TRAP, 23-MATCHES, NUMBER | BHB-8K | $10.00 |
| H8-BG2 | SNARK, REVERSE, STARS, TAXMAN FIPFOP, SLOTS, HURKLE, MUGWUMP | BHB-8K | $10.00 |
| | Machine Language Games (Front Panel) | | |
| H8-MG1 | Manipulation Games | MEM-4K | $10.00 |
| H8-MG2 | Number Games | MEM-4K | $10.00 |
| H8-MG3 | Casino Games | MEM-4K | $10.00 |
| H8-MG4 | Calculator & Clock | MEM-4K | $10.00 |
| | Machine Language Programs | | |
| H8-CODE | Ham Code Practice Program (Front Panel & Terminal Versions) | MEM-4K TRM-4K | $10.00 |
| H8-KING | Kingdom Game | TRM-4K | $10.00 |
| H8-DISM | Program Disassembler | TRM-4K | $10.00 |
| H8-EDUC | Educator-8080 Teaches machine programming interactively | TRM-4K | $10.00 |
| RADIO SHACK® TRS-80 SOFTWARE | | | |
| | Level One Basic Games and Programs | | |
| R8-BG1 | BIORYTHM, CHOMP, TEASER, BATUM CRAPS, TRAP, 23-MATCHES, NUMBER | L1B-4K | $10.00 |
| R8-BG2 | SNARK, REVERSE, STARS, TAXMAN FIPFOP, SLOTS, HURKLE, MUGWUMP | L1B-4K | $10.00 |

LEGEND:
BHB-8K = H8+8K+H9/H36 Uses Benton Harbor Basic
MEM-4K = H8+4K ONLY Uses Front Panel
TRM-4K = H8+4K+H9/H36 Interactive thru Terminal
L1B-4K = TRS-80+4K Uses Level I Basic

Includes: Cassette, Source Listing, and User Documentation
Terms: Check, Money Order, BAC/VISA, Master-Chg - NO CODs! Foreign orders in US funds, plus $1US/Cassette Postage
Shipment: In sequence as received by UPS or 1st Class Mail
Dealer Inquiries Invited; Program Contributions Solicited
Highest Royalties Paid

Multi-Micro Media Corp., P.O. Box 1025, Arvada, Colo. 80001

M38

---

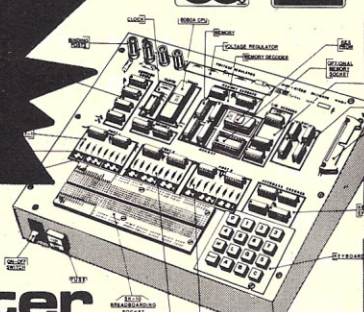
# DESIGN YOUR OWN INTERFACE WITH "KITTATURE"™


from **computer depot**

Now you can get the kits . . . and the manuals . . . to train yourself in Microprocessor Hardware development Computer Depot's exclusive "Kittature" package includes The MMD-1 Microcomputer system — which permits you to design your own interface in 10-20 minutes — and E & L BUGBOOKS V & VI with over 60 experiments.

master charge VISA

**FREE** BugBooks with the purchase of the MMD-1S in kit or assembled form. Only $600, assembled; $422.80 kit. Books also available individually for just $19.00 and $9.50 each.



**computer depot, inc.**
"Making Efficiency Economical"
3515 West 70th Street, Minneapolis, MN 55435
Phone: (612) 927-5601

Other educational "Kittature" is also available. Send for your FREE catalog from Computer Depot today!

```
010B  E0 01    C2   SUB B  1,X
010D  A2 00         SBC A  X
010F  25 05         BCS    C5
0111  7C A01E       INC    SAVEA
0114  20 F5         BRA    C2
0116  EB 01    C5   ADD B  1,X
0118  A9 00         ADC A  X
011A  36            PSH A
011B  FF A021       STX    SAVEX1
011E  FE A01F       LDX    SAVEX
0121  B6 A01E       LDA A  SAVEA
0124  A7 07         STA A  7,X       SAVE BCD TEMPERATURE
0126  8B 30         ADD A  #$30
0128  A7 00         STA A  X         SAVE ASCII TEMPERATURE
012A  32            PUL A
012B  08            INX
012C  FF A01F       STX    SAVEX
012F  FE A021       LDX    SAVEX1
0132  08            INX
0133  08            INX
0134  BC A01C       CPX    OUTSTR
0137  26 CF         BNE    C1
0139  C6 04         LDA B  #4
013B  7D A023       TST    MODE
013E  26 04         BNE    C4
0140  E7 07         STA B  7,X
0142  20 08         BRA    C8
               *    SHIFT TENTHS DIGIT & INSERT DECIMAL POINT
0144  A6 06    C4   LDA A  6,X
0146  A7 07         STA A  7,X
0148  86 2E         LDA A  #'.
014A  A7 06         STA A  6,X
014C  08       C8   INX
014D  08            INX
               *           REMOVE LEADING ZEROS
014E  A6 00    C6   LDA A  X
0150  81 30         CMP A  #'0
0152  26 08         BNE    C7
0154  86 20         LDA A  #$20
0156  A7 00         STA A  X
0158  08            INX
0159  5A            DEC B
015A  26 F2         BNE    C6
015C  39       C7   RTS
                    END
```

bly only work on that version because of modifications required for calling a machine-language subroutine. Follow these instructions carefully and in sequence.

1. Relocate the temperature-sensing subroutine starting at memory location 1EAF. Use the utility program supplied for this. Enter the following data into memory to set it up for relocation to where it can be called by BASIC:

| Address | Data |
|---|---|
| A02A | 0000 |
| A02C | 015C |
| A02E | 1EAF |
| A048 | A014 |

Execute the relocation program.

2. Save memory locations 1EAF to 200B on tape using the MIKBUG P command.
3. Load 8K BASIC version 2.0 into the computer. Do not execute it yet.
4. Load the version of the temperature-sensing subroutine that starts at address 1EAF.
5. Enter 2010 into memory locations 014E and 014F.
6. Enter G and execute BASIC.
7. Load the BASIC calling program.
8. Return to MIKBUG control either by RESET or PATCH.
9. Enter 1EAF into memory locations 0067 and 0068.
10. Enter G to execute BASIC.
11. Run the BASIC calling program. You will get both the temperature displays from the subroutine and the BASIC program. The first is an indication that you are entering the subroutine. If not, repeat steps 8 through 11.
12. To eliminate the display from the subroutine, return to MIKBUG and enter 39 in memory location 1F3B. Enter G and run the BASIC program.
13. If you restart BASIC from address 0100 or execute the BASIC commands NEW or LOAD, the data in 0067 and 0068 is changed and step 9 must be repeated.

The BASIC calling pro-

gram is in a loop. To make another temperature measurement just hit the return key. Control C will end the loop and return to READY.

### Calibration

The following procedure can be used if your thermometer does not correspond to the actual temperature because of the tolerance on the electrical components. It can also be used for calibrating the thermometer to the Celsius scale or for any other application you may have for this analog-to-digital converter technique.

This temperature-sensor system has a range from 0 to 120°F. It works by loading a constant into the index register and decrementing it until interrupted. The remaining value is then multiplied and divided by integer constants to obtain a value for temperature to the one-tenth degree. These constants are hexadecimal values stored in program bytes 2 through 5. They are labeled MULT, DIVIDE and INITX. For my setup, the multiplication constant is 24 or hex 18, the division constant is 44 or hex 2C, and the initial counter is 4618 or hex 120A.

Here is how to measure and compute these constants. The subroutine has two operating modes. In the temperature-sensing mode, it loads the constant into the index register and decrements it while the sensor is on. The calibration mode clears the index register and increments it. The calibration mode also bypasses the math routines and displays the actual number of loops executed during the sensor cycle. See subroutine lines 57 to 65. To enter the calibration mode, change the INC A instruction in the calling program to a NOP by changing the value in memory location A04D from 4C to 01.

The materials needed for calibration are a glass of water at room temperature, a container, preferably insulated, with ice and water, and a thermometer. Stir the water to equalize the temperature. The water will reach a temperature of 32°F.

Set up the calling program for the calibration mode. Dip the thermistor into the warm water. Note the temperature of your standard thermometer. Execute the calling program and note the loop count from the calibration mode displayed on your terminal. My values were 80° and 3165. Do the same with the ice water. Remember to stir the ice first and do not let the thermistor touch the ice or the side of the container when you are recording data. The ice could still be colder than 32°F. My loop count at 32°F was 4036. You may

```
                         NAM    CALLSUB
           0000   BEGSUB EQU    $0000
           A023   MODE   EQU    $A023
           E0E3   CONTRL EQU    $E0E3
      A048               ORG    $A048
      A048 A04A          FDB    $A04A
      A04A 34     ENTER  DES
      A04B 34            DES
      A04C 4F            CLR A
      A04D 4C            INC A          USE NOP FOR CALIBRATION
      A04E B7 A023       STA A  MODE
      A051 BD 0000       JSR    BEGSUB  MEASURE TEMPERATURE
      A054 7E E0E3       JMP    CONTRL
                         END
```

*Program 2. Program to call subroutine.*

```
                          NAM   RELOCATE
          E0E3     CONTRL EQU   $E0E3
          A02A     BEGMOV EQU   $A02A     ADDRESS FOR START OF PROGRAM
          A02C     ENDMOV EQU   $A02C     ADDRESS FOR END OF PROGRAM
          A02E     MOVETO EQU   $A02E     NEW STARTING ADDRESS
    A014                  ORG   $A014
    A014  FE A02A  ENTRY  LDX   BEGMOV
    A017  BE A02E         LDS   MOVETO
    A01A  A6 00    MOV1   LDA A X
    A01C  36              PSH A
    A01D  BC A02C         CPX   ENDMOV
    A020  27 05           BEQ   MOV2
    A022  31              INS
    A023  31              INS
    A024  08              INX
    A025  20 F3           BRA   MOV1
    A027  7E E0E3  MOV2   JMP   CONTRL
                          END
```

*Program 3. Program to relocate subroutine.*

wish to repeat this process a few times and average the data. The reason for having the glass of water at room temperature is that if you tried to measure the room temperature after you wet the thermistor in the ice water, the reading would be a wet-bulb temperature, colder than room temperature and dependent on the humidity. You risk breaking the thermistor leads if you dry it with a towel. Also, the sensor is very sensitive and would be affected by air currents, lights and your body heat.

You will notice that the loop count is higher at the lower temperature. This is because the thermistor increases resistance as the temperature decreases. At a higher resistance, the capacitor takes longer to charge. That is why the subroutine decrements the counter when measuring temperature and why it is important to carefully determine the initial value of the counter. My calibration tests produced the data in Example 1, which gives 871/48, or 18.14 counts per degree F, and makes it possible to measure to 0.1°F with precision.

Now we must do a few tricks because the math routines are only 16-bit integer precision; an intermediate value cannot exceed a decimal value of 65,536. First, the temperature difference must be an even number so it can be divided evenly by two. To get this difference, you may have to change the warm-water temperature. Then, round off the calibration count differential to a number evenly divisible by 20, and divide by 10 to shift the tenths of a degree into an integer. This is equivalent to multiplying the calibration constant by 10. For example, 871 rounded becomes 880, and 880/10 becomes 88.

Now make a ratio of the two measurement differences, divide each by 2 and convert to hexadecimal as in Example 2. Thus, 18 is the multiplication constant and 2C is the division constant. Load the multiplication constant into the MULT value at subroutine byte 2 and the divide constant into the DIVIDE value at subroutine byte 3.

Why is it necessary to divide the numerator by 2? Remember that there are approximately 18.14 counts per degree F. At 80°F, the index register would have a decimal value of 18.14 X 80 = 1451. Mulitply 1451 by the multiplication constant 48, and the result is 69,648. This exceeds the 65,536 limit for 2 bytes. By reducing the multiplication constant by 2, the maximum value obtainable at 120°F is 18.14 X 120 X 24 = 52,243.

If you will be satisfied with a thermometer range of 32°F to 120°F, you can convert the calibration count for 32°F (4036) to hex (0FC4) and load it into the INITX value at subroutine bytes 4 and 5. In this case, a constant 320 (for 32 X 10), hex value 0140, must be added to the resultant measurement. This is done in lines 149 and 150 (see Example 3).

To extend the measurable range from 0°F to 120°F, the INITX value must be computed using the equation in Example 4.

Now load this value into the INITX value in subroutine bytes 4 and 5. Lines 149 and 150 must now remain as in the listing. Remember, you must change the value in memory location A04D from 01 to 4C to return the program to the thermometer mode.

## The Next Step

This article has provided a useful tool for your computer. By itself, the temperature sensor does not do much. It must become a part of your special application. This could be simply to monitor and record the outside temperature every hour. But its real value is in an application for temperature control, especially for solar-heating equipment. Now you can get started on your own special project. ■

```
0010 REM PROGRAM TO UTILIZE TEMPERATURE SUBROUTINE
0020 REM BY JOHN P. BAUERNSCHUB, JR.
0030 REM JULY, 1977
0040 B = 40984
0050 H = 40995
0060 G = 1
0070 POKE( H,G)
0080 A = USER(X)
0090 C = PEEK(B)*256
0100 D = PEEK(B+1)
0110 T = (C+D)/10
0120 PRINT
0130 PRINT "THE TEMPERATURE IS"
0140 PRINT T;"DEGREES F"
0150 INPUT A$
0160 GOTO 80
```

*Program 4. BASIC program to call subroutine.*

| | Temperature | Loop count |
|---|---|---|
| | 80° | 3165 |
| | 32° | 4036 |
| Absolute value of difference | 48 | 871 |

*Example 1.*

$$\frac{\text{Temperature difference}}{\text{Counter difference/ 10}} = \frac{48}{88} = \frac{24}{44} = \frac{18_{16}}{2C_{16}}$$

*Example 2.*

| Line | Change | To |
|---|---|---|
| 149 | CB 00 ADD B #0 | CB 40 ADD B #$40 |
| 150 | 89 00 ADC A #0 | 89 01 ADC A #$01 |

*Example 3.*

$$\text{Loop count for } 32^\circ\text{F} + \frac{32 \times 10}{\dfrac{\text{Multiplication constant}}{\text{Divide constant}}}$$

For example:

$$4036 + \frac{32 \times 10}{\dfrac{24}{44}} = 4036 + 587 = 4623 = 120F_{16}$$

*Example 4.*

# A Different Approach to HI-LO

## *let the computer guess!*

*Irwin Doliner*
*PO Box 290*
*Owings Mills MD 21117*

```
RUN

LET'S SEE IF EITHER OF US CAN READ THE OTHER'S MIND. THIS IS
A NUMBER GUESSING GAME. ONE OF US PICKS A NUMBER BETWEEN THE
LO AND HI VALUE AND THE OTHER TRIES TO GUESS IT IN A NUMBER
OF GUESSES WHICH IS REASONABLE. YOU MAY DECIDE WHICH OF US
PICKS THE NUMBER. THE OTHER ONE WILL DO THE GUESSING.

LO,HI? 1,100
WOULD YOU LIKE TO PICK THE NUMBER FOR ME TO GUESS? N
O.K. THEN I WILL PICK A NUMBER AND YOU WILL GUESS.
I WILL ASK FOR YOUR FIRST GUESS WHEN I HAVE THOUGHT OF MY
NUMBER. (THINK, THINK, THINK, THINK, THINK, O.K. I HAVE IT.)
YOUR GUESS NUMBER 1 ? 50
50     IS HIGH! YOUR GUESS NUMBER  2 ? 25
25     IS LOW! YOUR GUESS NUMBER  3 ? 37
37     IS HIGH! YOUR GUESS NUMBER  4 ? 31
31     IS HIGH! YOUR GUESS NUMBER  5 ? 28
28     IS HIGH! YOUR GUESS NUMBER  6 ? 26
RIGHT!THAT WAS SOME TERRIFIC GUESSING!
WOULD YOU LIKE TO PLAY AGAIN ? Y
LO,HI? -100,100
WOULD YOU LIKE TO PICK THE NUMBER FOR ME TO GUESS? Y
HAVE YOU PICKED YOUR NUMBER YET? Y
YOU MUST TELL ME IF MY GUESS IS 'HIGH', 'LOW', OR 'RIGHT'.
GUESS NUMBER      1      IS      0  ?  H
GUESS NUMBER      2      IS    -51  ?   H
GUESS NUMBER      3      IS    -76  ?   L
GUESS NUMBER      4      IS    -64  ?   R
***** I THINK I DID THAT PRETTY WELL! *****
WOULD YOU LIKE TO PLAY AGAIN ? Y
LO,HI? 300,500
WOULD YOU LIKE TO PICK THE NUMBER FOR ME TO GUESS? Y
HAVE YOU PICKED YOUR NUMBER YET? Y
YOU MUST TELL ME IF MY GUESS IS 'HIGH', 'LOW', OR 'RIGHT'.
GUESS NUMBER      1      IS     400  ?  H
GUESS NUMBER      2      IS     349  ?  H
GUESS NUMBER      3      IS     324  ?  H
GUESS NUMBER      4      IS     311  ?  H
GUESS NUMBER      5      IS     305  ?  H
GUESS NUMBER      6      IS     302  ?  L
GUESS NUMBER      7      IS     303  ?  L
GUESS NUMBER      8      IS     304  ?  L
YOU MUST PLAY FAIRLY OR THE GAME IS NO FUN.
EITHER YOU CHANGED YOUR NUMBER OR FORGOT WHAT IT WAS'

WOULD YOU LIKE TO PLAY AGAIN  ?   Y
LO,HI? -100,0
WOULD YOU LIKE TO PICK THE NUMBER FOR ME TO GUESS?  N
O.K. THEN I WILL PICK A NUMBER AND YOU WILL GUESS.
I WILL ASK FOR YOUR FIRST GUESS WHEN I HAVE THOUGHT OF MY
NUMBER. (THINK, THINK, THINK, THINK, THINK, O.K. I HAVE IT.)
YOUR GUESS NUMBER        1   ?   -50
-50     IS HIGH! YOUR GUESS NUMBER  2   ?  -75
-75     IS LOW! YOUR GUESS NUMBER   3   ?  -62
-62     IS HIGH! YOUR GUESS NUMBER  4   ?  -68
-68     IS HIGH! YOUR GUESS NUMBER  5   ?  -72
-72     IS LOW! YOUR GUESS NUMBER   6   ?  -70
-70     IS LOW! YOUR GUESS NUMBER   7   ?  -69
RIGHT! THAT WAS SOME TERRIFIC GUESSING!
WOULD YOU LIKE TO PLAY AGAIN ? N
```

*Sample run.*

```
RUN
WANT INSTRUCTIONS ? YES
I WILL SELECT A NUMBER BETWEEN 1 AND 100 AND YOU WILL TRY TO GUESS IT. AFTER EACH
WRONG GUESS I WILL TELL YOU IF YOU WERE HIGH OR LOW.

WHAT IS YOUR GUESS ? NOW JUST A COTTON-PICKIN MINUTE! WHEN DO I GET A TURN TO
SELECT A NUMBER?

THAT IS NOT THE WAY THE GAME GOES, PLEASE PLAY ACCORDING TO THE RULES.
WHAT IS YOUR GUESS ? I QUIT!
```

The above dialogue is representative of the way most number guessing programs work. The computer selects a number and the human must try to guess it. This one-sided arrangement is decidedly unfair, and obviously could not be allowed to continue. Being a fair-minded individual I have written a program that rectifies this inequity.

With GESNUMBR either player (human or computer) may select the number from any reasonable range of values (e.g., 1 to 100 or -200 to 400) while the other does the guessing. In addition the human designates the range and indicates which player will do the guessing.

The program uses an efficient algorithm, similar to the Binary Search algorithm, for guessing your number. The first guess is the number in the middle of the range. If this guess is incorrect the range has been effectively reduced to the upper or lower half range (depending upon whether the guess was low or high respectively). This process is repeated until the number is found. If the initial range is X numbers long (i.e., High - Low + 1 = X) then the program will find the number in N guesses or less where $2^{N-1} <= X < 2^N$.

The sample run shows how the program applies this algorithm to finding the number. When you are doing the guessing the computer computes N as indicated above from the specified range. A different complimenting message is generated depending upon how many guesses you required to find the number. However after 3*N incorrect guesses you are told the answer.

Have fun with the game but remember, play fairly; the computer knows when you are cheating. ■

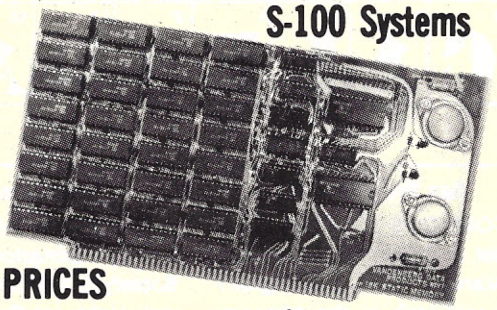
*Program listing.*

```
0     REM      PGM:          GESNUMBR
1     REM      WRITTEN BY:   IRV DOLINER (INTERACTIVE DATA SYSTEMS)
2     REM      ADDRESS:      P. O. BOX 290
3     REM                    OWINGS MILLS, MARYLAND 21117

5     PRINT:PRINT:PRINT
9     PRINT"LET'S SEE IF EITHER OF US CAN READ THE OTHER'S MIND. THIS IS"
10    PRINT"A NUMBER GUESSING GAME' ONE OF US PICKS A NUMBER BETWEEN THE"
20    PRINT"LO AND HI VALUE AND THE OTHER TRIES TO GUESS IT IN A NUMBER"
30    PRINT"OF GUESSES WHICH IS REASONABLE. YOU MAY DECIDE WHICH OF US"
40    PRINT"PICKS THE NUMBER. THE OTHER ONE WILL DO THE GUESSING."
41    PRINT
42    INPUT"LO,HI";L0,H1
43    IF LO<H1 GOTO46
44    PRINT"LO MUST BE <HI"
45    GOTO 42
46    R0=H1-L0+1
47    T9=0
48    I=1
49    IF I>=R0 GOTO 54
50    T9=T9+1
51    I=I*2
52    GOTO 49
54    PRINT"WOULD YOU LIKE TO PICK THE NUMBER FOR ME TO GUESS";
55    INPUT A$
60    IF LEFT$(A$,1)="Y" THEN 140
70    IF LEFT$(A$,1)<>"N" THEN 100
80    PRINT"O.K. THEN I WILL PICK A NUMBER AND YOU WILL GUESS."
90    GOTO 180
100   PRINT"I DON'T UNDERSTAND YOUR ANSWER."
110   GOTO50
120   INPUT"HAVE YOU PICKED YOUR NUMBER YET";A$
130   IF LEFT$(A$,1)="Y" GOTO 150
140   FOR I=1TO1000:NEXTI:GOTO 120
150   PRINT"YOU MUST TELL ME IF MY GUESS IS 'HIGH', 'LOW', OR 'RIGHT'."
155   PRINT
160   GOSUB20000
170   GOTO 210
180   PRINT"I WILL ASK FOR YOUR FIRST GUESS WHEN I HAVE THOUGHT OF MY"
190   PRINT"NUMBER. (";
195   FOR I=1TO5:FORJ=1TO200:NEXTJ:PRINT"THINK, ";:NEXTI
196   PRINT"O.K. I HAVE IT.)"
200   GOSUB10000
210   PRINT"WOULD YOU LIKE TO PLAY AGAIN";N$;
220   INPUT A$
230   IF LEFT$(A$,1)="N"GOTO270
240   IF LEFT$(A$,1)="Y"GOTO 42
250   PRINT"I DON'T UNDERSTAND YOUR ANSWER. ";
260   GOTO210
270   GOTO 30000
900   REM <<<<<<<<<<THE HUMAN DOES THE GUESSING >>>>>>>>>>
10000   H=LO+INT(RO*RND(2))+1
10005   PRINT
10010   M=0
10020   M=M+1
10030   PRINT"YOUR GUESS NUMBER ' ;M;
10040   INPUT G
10050   IF G>HGOTO10180 10060
10060   IF G<HGOTO 10220
10065   PRINT"RIGHT!";
10070   IF M>T9 GOTO10100
10080   PRINT"THAT WAS SOME TERRIFIC GUESSING!"
10090   RETURN
10100   IF M>INT(T9*1.5+.5) GOTO 10130
10110   PRINT"VERY GOOD GUESS WORK!"
10120   RETURN
10130   IF M>2*T9 GOTO10150
10140   PRINT"NOT BAD GUESSING AT ALL!"
10150   RETURN
10180   PRINT:PRINTG;" IS HIGH! ";
```

```
10190  IF M<3*T9 GOTO 10020
10200  PRINT"THE CORRECT ANSWER IS ";H
10210  RETURN
10220  PRINT:PRINTG;" IS LOW! ' ;
10230  GOTO 10190
19000  REM <<<<<<<<<<< THE COMPUTER DOES THE GUESSING >>>>>>>>>>
20000  H=H1
20010  L=L0
20020  N=0
20025  N=N+1
20030  G=INT((H+L)/2)
20040  PRINT "GUESS NUMBER";N;"IS";G;
20050  INPUT A$
20060  IF LEFT$(A$,1)="H" GOTO 20115
20070  IF LEFT$(A$,1)="L" GOTO 20135
20080  IF LEFT$(A$,1)="R" THEN 20220
20090  PRINT "I DON'T UNDERSTAND YOUR ANSWER, WAS I 'HIGH', 'LOW',"
20100  PRINT"OR 'RIGHT' ";
20110  GOTO 20050
20115  IF L>=H GOTO 20190
20120  H=G-1
20125  IF G<L0 GOTO 20190
20130  GOTO 20025
20135  IF L>=H GOTO 20190
20140  L=G+1
20145  IF G>=H1 GOTO 20190
20160  GOTO 20025
20180  REM <<<<<<<<<<< THE HUMAN HAS CHANGED HIS/HER NUMBER >>>>>>>>>>>
20190  PRINT"YOU MUST PLAY FAIRLY OR THE GAME IS NO FUN."
20200  PRINT"EITHER YOU CHANGED YOUR NUMBER OR FORGOT WHAT IT WAS."
20210  RETURN
20220  X=X+1
20221  IF X>4 THEN X=X-4
20225  PRINT:PRINT:PRINT:PRINT:PRINT:PRINT""
20230  ON X GOTO 20240,20260,20280,20300
20240  PRINT"***** I THINK I DID THAT PRETTY WELL! *****"
20250  GOTO 20310
20260  PRINT"***** BOY! AM I JUST THE GREATEST! *****"
20270  GOTO 20310
20280  PRINT"***** I AM SO TERRIFIC I JUST LOVE MYSELF TO PIECES! *****"
20290  GOTO 20310
20300  PRINT"***** I THINK I AM SMARTER THAN MOST HUMANS—EXCEPT YOU OF COURSE! *****"
20310  PRINT:PRINT:PRINT:PRINT:PRINT
20320  RETURN
30000  END
```

# 16K Static RAM

## S-100 Systems



**NEW LOW PRICES**

**$330 Kit** **$365 Assembled**

- Very Low Power—650MA+5V; 90MA+12V; 16MA-5V
- Applications Notes—6800 and 6502 Sys.
- Low-profile sockets for all chips
- Solder mask; silk screen; plated through holes
- Each 4K addressable to any 4K boundary
- Fully buffered S-100 bus—gold-plated contacts
- NEC $\mu$PD 410 D memories

**COD, Master Charge, B of A, Visa Accepted . . . . Orders shipped prepaid. California residents add 6% sales tax.**

**VANDENBERG DATA PRODUCTS**

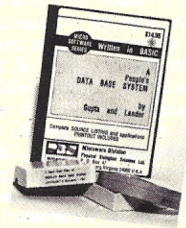
PO BOX 2507
SANTA MARIA, CALIFORNIA 93454 805-937-7951 V14

Discover the power of DATA BASE approach to software on your micro.

# PEOPLE'S DATA BASE SYSTEM

**BASIC STATEMENTS NEEDED**

| | |
|---|---|
| DATA | ON GOSUB |
| DIM | PRINT |
| FOR | READ |
| GOSUB | REM |
| IF | RETURN |
| INPUT | STRING VARIABLES with MID $ |
| NEXT | |



PDBS is the most powerful program offered to the owner of a personal computer to date. DATA BASE commands whose source codes are fully listed (plenty of examples and flowcharts) are:

| | | |
|---|---|---|
| ADD | COVARIANCE | NSORT |
| AVERAGE | DELETE | RESTORE |
| CLIST | LABELS | SUM |
| COMPRESS | LIST | VARIANCE |

Suggestions for program modification are inter-mixed should you have a more powerful BASIC. Some of the many advantages of the DATA BASE approach to software are: #1, no duplication of effort in writing the same program step for different programs; #2, ideal for on-line inquiry and expansion into more powerful programs. You do not need to know programming. Just type the program in and RUN.

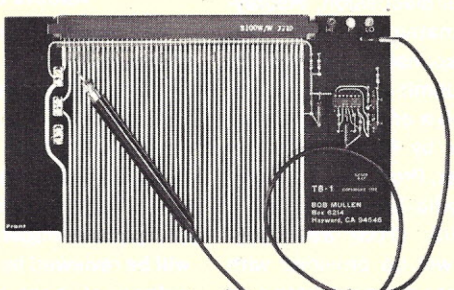
Price: $14.95
Add $5.00 for paper tape copy
Mst. Chg. & VISA accepted (include all information on card).

Send Orders to: *MICROWARE DIVISION*
*Physical Biological Sciences, Ltd.*
*P.O. Box 47*
*Blacksburg, Virginia 24060* M37

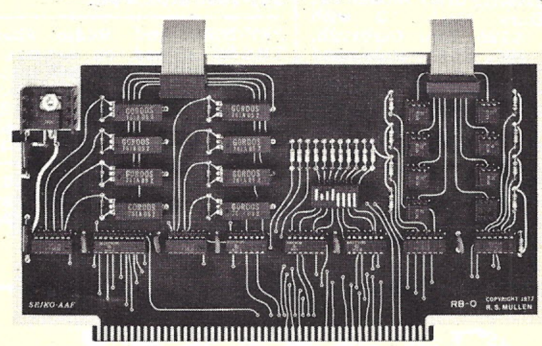
# "Products that make your computer useful"

# EXTEND... CONTROL!

CAVE grafx



Whether for troubleshooting or analysis, if you have an S-100 machine at some point you will need our Extender Board with Logic Probe Kit (**$35**). The logic probe makes it easy to see which signals are going where . . . our special edge connector provides easy clip lead probing, jumper links in supply lines allow for fusing/current measurement/shut-down independent of system, and a non-skid needlepoint probe helps prevent accidental shorting. As with other Mullen kits, you also have quality parts, detailed instructions, and a realistic price.



The Altair/S-100 compatible Relay/Opto-Isolator Control Board Kit (**$117**) is a natural for controlling audio systems, time lapse photography experiments, model trains, robot devices, or any application where you need a number of intelligent switches . . . more uses are discovered daily, as detailed in our applications notes. 8 reed relays respond to an 8 bit word from your computer; 8 opto-isolators accept an 8 bit word from the outside world and send it back to your machine for handshaking or further control purposes. Includes detailed instructions.

# MULLEN COMPUTER BOARDS

BOX 6214, HAYWARD, CA 94545 M32

Available by direct mail (shipped ppd. in USA from stock; Cal res add tax) or at many fine computer stores. Dealer inquiries invited.

# Kilobaud Classified

Kilobaud classified advertisements are intended for use by those individuals desiring to buy, sell, or trade used computer equipment or software. No commercial ads are accepted.

Two sizes of ads are available. The $5 box allows five lines of about 22 characters each, including spaces and punctuation. The $10 box provides ten lines of type – again, each line is about 22 characters. Minimize capital letters, as they use twice the space of small characters. Payment is required in advance with ad copy. We cannot bill, or accept credit. Oversize ads are not accepted. Each subscriber is limited to two (2) identical ads in any given issue.

Advertising text and payment must reach us 60 days in advance of publication. For example, advertising copy for the March issue (mailed in February) must be in our hands on January 1. The publisher reserves the right to refuse a questionable or not applicable advertisement. Mail advertisements to: KILOBAUD CLASSIFIED, Kilobaud, Peterborough, NH 03458. Do not include any other material with your ad, as it may be delayed.

---

KSR-35 TTY: Pin feed, less than 500 hours, complete documentation, excellent condition, Bonus: 2 Stelma modems. $700 + shipping. S.T. Stolen, 2920 N. Armistead Ave., Hampton VA 23666. (804) 827-7311 after 6 pm.

---

PET users need info? Getting started with your PET workbook, $4. Workbook has helpful info, sample programs, exercises and fundamental features of PET BASIC. Send SASE for details. D. Smith, Box 921, Los Alamos NM 87544.

---

CUTS fix: Unacceptable error rate with otherwise functional board is probably not your fault! Send $2 for copyrighted doc. of board mods. and explanation of the problem. DIAS 234 Union St., Schenectady NY 12305.

---

PerSci Intelligent Disk Controller & S100 adapter w. resident monitor, $700. G. Lyons, 280 Henderson, Jersey City NJ 07302. (201)451-2905.

---

Sell: Teletype Model 33KSR rebuilt, $375; Teletype Model 33ASR used, good cond., $675; Sperry cassette drive model TT, $40; Dura Mach 10 with manual, $225. Lou Carbaugh, PO Box 398, New Cumberland PA 17070.

---

For Sale: 8K EMPL 1.0 (a micro APL for the Z-80/8080) with user's manual, $10 on Tarbell cassette; $20 on Paper Tape, North Star Disk, CUTS cassette or Mits cassette. Erik T. Mueller, 36 Homestead La., Roosevelt NJ 08555. (609) 448-2605.

---

Conductive Foam. ¼" thick, 3¢/sq. in. plus 25¢ postage. WF, 713 George Ln., Glendora NJ 08029.

---

For Sale: IBM 1401 card sys. 4K 1401, 1402 and U750 prt. EAI TR-10 analog comp. Sanders TV-type terminal. Box 7281, Baltimore MD 21218, (301) 366-4662.

---

For Sale: Altair 8800 with 12K RAM and 18-slot motherboard with 6 edge connectors. $500. Evenings, (404) 633-3569.

---

New Poly Software: Using A00 BASIC & dynamic graphics-Bomber; Missile; Equation Plot; Biorhythm; Tank; Pursuit; Lunar Lander; Sketch; Splat. $3 each. Send for complete list. Ted Carter, 902 Pinecrest, Richardson TX 75080, or call (214) 235-0915.

---

Custom KIM-1 in Viatron mainframe & cabinet w/kbd, 2 digital cassette decks, 5K RAM, 5 S-100 sockets, D/A cnvtr., ps, Tiny BASIC, minidisassembler, chess, much more software. $545 or best offer. Paul Peckins, 52 Monument Ave., Charlestown MA 02129. (617) 241-7629 after 6 pm.

---

PET-2001 and Radio Shack TRS-80 arrived on campus. I want to survey users and report results to any interested hobbyists. Write: Professor Bill Parks, Walters State Community College, Morristown TN 37814.

---

Microdata 1600 CPU (Reality) — Wanted for above: mag tape controller & disc controller. (201) 751-3000.

---

# Contest!

The votes are still pouring in in our contest; and the winner for December is David Yulke, author of "File Structures Simplified (*Kilobaud* No. 12, page 106). Kudos to you, Dave Y.

Winner of a choice of any book from the Book Nook is David Smith of North Vancouver BC Canada, whose card was selected in a drawing of all those submitted. Congrats to you, too, Dave S.

We're still counting all the votes for 1977, and will be announcing the winner for the year soon.

Keep voting!

# KB CALENDAR

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | K | B | | | | | |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| C | A | L | E | N | D | A | R |

### San Francisco CA

COMPCON '78, Jack Tar Hotel, San Francisco, February 28-March 2, 1978, (starting at 7 pm) will look at the phenomenon of personal computing. Four panel sessions have been arranged with experts who will be discussing various aspects of the computer revolution. The conference registration fee covers attendance at all personal computing sessions and exhibits. There is a registration fee of $5 for individuals wishing to only attend the personal computing sessions and exhibits. Contact organizers Alice Ahlgren, Mountain View CA, (415) 964-7400 and/or Robert Albrecht, Menlo Park CA, (415) 323-6117, for more information.

### Washington DC

Amateur Computing 78 microcomputer festival will be held July 22-23 at the Sheraton National Motor Hotel, Arlington VA, overlooking Washington DC. Those interested in presenting a paper, participating in a panel discussion, displaying an amateur computer system or sponsoring a tutorial should submit a letter of intent along with a one-page abstract or outline by April 15 to John Wall Miller, Program Chairman, 6921 Pacific Lane, Annandale VA 22003, (703) 256-5702. Authors will be provided with instructions for preparation of camera-ready papers, which are due by June 1.

Areas of interest are: personal computing; amateur radio and microcomputers; home educational uses of computers; speech, music and graphics; standards for hardware, software and interfacing to the real world; and subjects of interest to beginners.

For information, write AMRAD, PO Box 682, McLean VA 22101.

### Long Beach CA

PERCOMP '78 (co-sponsored by the International Computer Society/SCCS and the Rockwell Hobbyist Computer Club) will be held at the Long Beach Convention Center, Long Beach CA, April 28-30, 1978. PERCOMP is a selling show designed with the home computerist and small-business person in mind. For information concerning seminars, contact James Lindwedel, Technical Program Chairperson, PERCOMP '78, 1833 E. 17th St., Santa Ana CA 92701.

### Lexington VA

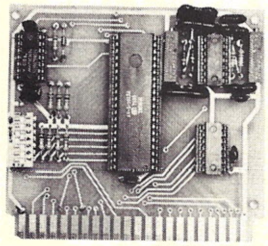
A two-week course in digital-electronics and microcomputer-interfacing fundamentals will be held at Virginia Military Institute from July 17 through July 29, 1978. For information and registration forms write to: Dr. Philip B. Peters, Dept. of Physics, VMI, Lexington VA 24450.

### Atlanta GA

Papers are invited for presentation at the 16th Annual Convention of the Association for Educational Data Systems, Atlanta GA, May 15-19, 1978. Papers are solicited in all categories of educational use of computers. Papers submitted will be reviewed by a panel, and authors of those accepted will be invited to present their papers and to have them published in the proceedings.

Judges will select an outstanding paper from each category, and a panel will select a best paper on the basis of content, presentation and overall quality to receive a $500 cash award. For further information, contact: Dr. James E. Eisele, Office of Computing Activities, University of Georgia, Athens GA 30602.

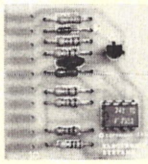
# ELECTRONIC SYSTEMS
## p.o. box 212 · Burlingame CA 94010
## (408) 374-5984



### UART & BAUD RATE GENERATOR

Part no. 101
- Converts serial to parallel and parallel to serial
- Low cost on board baud rate generator
- Baud rates: 110, 150, 300, 600, 1200, and 2400
- Low power drain +5 volts and -12 volts required
- TTL compatible
- All characters contain a start bit, 5 to 8 data bits, 1 or 2 stop bits, and either odd or even parity.
- All connections go to a 44 pin gold plated edge connector
- Board only $12.00; with parts $35.00

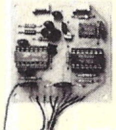
### RS-232/TTL INTERFACE



Part no. 232
- Converts TTL to RS-232, and converts RS-232 to TTL
- Two separate circuits
- Requires -12 and +12 volts
- All connections go to a 10 pin gold plated edge connector
- Board only $4.50; with parts $7.00

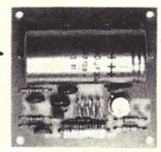
### DC POWER SUPPLY

Part no. 6085
- Board supplies a regulated +5 volts at 3 amps., +12,-12, and -5 volts at 1 amp.
- Board has filters, rectifiers, and regulators
- Power required is 8 volts AC at 3 amps., and 24 volts AC C.T. at 1.5 amps.
- Board only $12.50



### TAPE INTERFACE

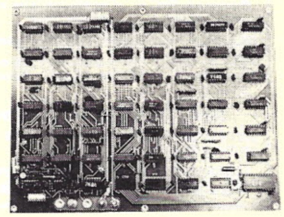
Part no. 111
- Play and record Kansas City Standard tapes
- Converts a low cost tape recorder to a digital recorder
- Works up to 1200 baud
- Digital in and out are TTL-serial
- Output of board connects to mic. in of recorder
- Earphone of recorder connects to input on board
- Requires +5 volts, low power drain
- Board $7.60; with parts $27.50
- No coils



### RF MODULATOR

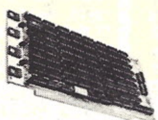
Part no. 107
- Converts video to AM modulated RF, Channels 2 or 3
- Power required is 12 volts AC C.T., or +5 volts DC
- Board $4.50; with parts $13.50

### TELEVISION TYPEWRITER



Part no. 106
- Stand alone TVT
- 32 char/line, 16 lines, modifications for 64 char/line included
- Parallel ASCII (TTL) input
- Video output
- 1K on board memory
- Output for computer controlled curser
- Auto scroll
- Non-distructive curser
- Curser inputs: up, down, left, right, home, EOL, EOS
- Scroll up, down
- Requires +5 volts at 1.5 amps, and -12 volts at 30 mA
- Board only $39.00; with parts $145.00

### 8K STATIC RAM



Part no. 300
- 8K Altair bus memory
- Uses 2102 Static memory chips
- Memory protect
- Gold contacts
- Wait states
- On board regulator
- S-100 bus compatible
- Vector input option
- TRI state buffered
- Board only $22.50; with parts $160.00

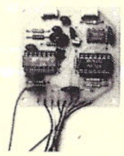
### TIDMA

Part no. 112
- Tape Interface Direct Memory Access
- Record and play programs without bootstrap loader (no prom) has FSK encoder/decoder for direct connections to low cost recorder at 625 baud rate, and direct connections for inputs and outputs to a digital recorder at any baud rate.
- S-100 bus compatible
- Comes assembled and tested for $160.00

### APPLE I MOTHER BOARD

Part no. 102
- 10 slots – 44 pin (.156) connectors spaced ¾" apart
- Connects to edge connector of computer
- Pin 20 and 22 connects to X & Z for power and ground
- Board has provisions for bypass capacitors
- Board cost $15.00

### MODEM



Part no. 109
- Type 103
- Full or half duplex
- Works up to 300 baud
- Originate or Answer
- No coils, only low cost components
- TTL input and output-serial
- Connect 8 ohm speaker and crystal mic. directly to board
- Uses XR FSK demodulator
- Requires +5 volts
- Board $7.60; with parts $27.50

### To Order:

Mention part number and description. For parts kits add "A" to part number. Shipping paid for orders accompanied by check, money order, or Master Charge, BankAmericard, or VISA number, expiration date and signature. Shipping charges added to C.O.D. orders. California residents add 6.5% for tax. Parts kits include sockets for all ICs, components, and circuit board. Documentation is included with all products. Dealer inquiries invited. 24 Hour Order Line: (408) 374-5984.

VISA master charge

E21

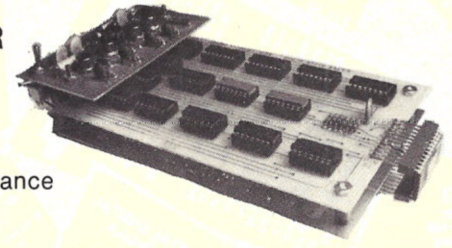
| VISIBLE OR INFRA RED | USED FOR CHARACTER RECOGNITION FOR COMPUTERS WITH EXTERNAL CIRCUITS | MAY BE USED IN A VACUUM, UNDER WATER, HIGH ALTITUDE | IN MAGNETIC ENVIRONMENT BECAUSE THERE IS NO HIGH VOLTAGE OR MAGNETIC DEFLECTION |
|---|---|---|---|

MINATURE SOLID STATE

# 202 VIDEO CAMERA KIT

**FEATURING A. . . 100 × 100 BIT SELF SCANNING CHARGED COUPLED DEVICE**

THIS UNIQUE UPDATED CAMERA KIT
FEATURES THE FAIRCHILD CCD 202C IMAGE SENSOR

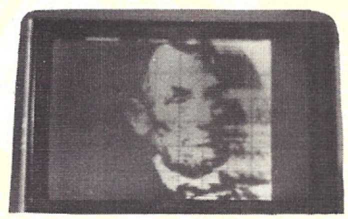


*ADVANTAGES*

- **IN THE FUTURE WE WILL SUPPLY A COMPUTER VIDEO INTERFACE CARD**
- All clock voltages operate at 6V reguiring no adjustments
- Higher video output signal
- We supply the power board, so only a 5V 1 Amp power source is needed
- The circuitry has been simplified for easier assembly
- Two level TTL output is supplied for interfacing

*FEATURES*

- Sensitive to infra red as well as visible light
- May be used for IR surveillance with an IR light source
- Excellent for standard surveillance work, because of light weight and small size
- All components mounted on parallel 3¾″ × 6½″ single sided boards
- Total weight under 1 lb.



We supply all semiconductors, boards, data sheets, diagrams, resistors and capacitors, and 8MM lens.
Sorry we do not supply the case, batteries and 5V supply.

**$349.00**

Add $2.00 Postage and Handling

### UNIVERSAL 4Kx8 MEMORY BOARD KIT $69.95

32-2102-1 fully buffered, 16 address lines, on board decoding for any 4 of 64 pages, standard 44 pin buss, may be used with F-8 & KIM

### EXPANDABLE F8 CPU BOARD KIT $99.00

featuring Fairbug PSU.1K-of static ram, RS 232 interface, documentation, 64 BYTE regis.er

### 4K BASIC FOR FAIRBUG F8

on paper tape . . . . . . . . . . $25.00

### C/MOS (DIODE CLAMPED)

| | | | |
|---|---|---|---|
| | 4012– .16 | 4023– .16 | 4046– 1.45 |
| | 4013– .29 | 4024– .66 | 4049– .34 |
| 4001– .16 | 4015– .74 | 4025– .16 | 4050– .34 |
| 4002– .16 | 4016– .29 | 4027– .37 | 4055– 1.25 |
| 4006– .95 | 4017– .78 | 4028– .73 | 4066– .58 |
| 4007– .16 | 4018– .78 | 4029– .77 | 4071– .24 |
| 4009– .37 | 4019– .37 | 4030– .33 | 4076– .97 |
| 4010– .37 | 4020– .84 | 4035– .97 | 74C10– .22 |
| 4011– .16 | 4022– .85 | 4042– .58 | 74C193 1.50 |

WSU-30 Hand wire wrap tool used to wrap, unwrap & strip # 30 wire $5.30

#24, EIGHT CONDUCTOR SPECTRA FLAT CABLE 10′/$1.50 100′/$13.50

### DIP SOCKETS

8 PIN – .22 24 PIN – .40
14 PIN – .25 28 PIN – .50
16 PIN – .28 40 PIN – .60
18 PIN – .30

#30 WIRE WRAP WIRE SINGLE STRAND 100′/$1.40

2708-8K EPROM (450 ns) $10.95
2522 STATIC SHIFT REG $ 1.95
2513 CHARACTER GEN $ 6.75
2518-HEX 32 BIT SR $ 3.50
TMS4050NL-4K dys RAM . . . . . . . $ 3.95
5203-2K EPROM . . . . . . . . . . $ 4.50
2102-1 1024 BT RAM (450 ns) $ .99
2107B . . . . . . . . . . . . . . . . . $ 3.50
MK4008P . . . . . . . . . . . . . . . . $ 1.95
5280 4K DYNAMIC RAM $ 4.75
1702A UV PROM . . . . . . . . . . $ 4.95
5204-4K PROM . . . . . . . . . . $10.95
82S23 . . . . . . . . . . . . . . . . . . .$ 1.95
AY-5-1013 UART . . . . . . . . . . $ 6.95

CTS 206-8 eight position dip switch . . . . $1.90
CTS-206-4 four position dip switch . . . .$1.45
LIGHT ACTIVATED SCR's TO-18, 200V 1A . . . . $ 1.10

### SILICON SOLAR CELLS

2¼″ diameter .4V at 500 ma $4.00

### RS232 CONNECTORS

DB 25P male . . . . . $2.95
DB 25S female . . . . $3.50

### LED READOUTS

FND 359 C.C. .4″ $ .50
FCS 8024 - 4 digit C.C. 8″ display $5.95
FND 503 C.C. .5″ $ .85
FND 510 C.A. .5″ $ .85
DL-704-.3″ C.C. $ .95
MAN-7-.3″ C.A. $ .95
NS 33-3 dig. array $ .75
DL 747 C.A. .6″ $1.95

### PRINTED CIRCUIT BOARD

4-1/2″×6-1/2″ SINGLE SIDED EPOXY BOARD 1/16″ thick, unetched $.60 ea. . . . . . . . . 5/$2.60

### 7 WATT LD-65 LASER DIODE IR $8.95

2N 3820 P FET . . . . . . . . . . $ .45
2N 5457 N FET . . . . . . . . . . $ .45
2N2646 . . . . . . . . . . . . . . . $ .45
ER 900 TRIGGER DIODES . . 4/$1.00
2N 6028 PROG. UJT . . . . . . . $ .65

MINIATURE MULTI-TURN TRIM POTS 100, 500, 1K,2K,5K,10K,25K,50K,100K, 200K,500K 1 Meg. $.75 each. . . . . 3/$2.00

CCD201- charge coupled device 100 × 100 image sensor . . . . . . . . . . . . .$88.00

### VERIPAX PC BOARD

This board is a 1/16″ single sided paper epoxy board, 4½″x6½″ DRILLED and ETCHED which will hold up to 21 single 14 pin IC's or 8, 16, or LSI DIP IC's with busses for power supply connector. . . . . . . . $4.00

MV 5691 YELLOW-GREEN BIPOLAR LED . . . . . . . . . . . $ .90
FP 100 PHOTO TRANS . . . . . . . . $ .50
RED, YELLOW, GREEN or AMBER LARGE LED's . . . . . . . . . 6/$1.00
1L-5 (MCT-2) OPTO-ISOLATOR . . . $ .75
MOLEX PINS . . . . . . . . . . . 100/$1.00 1000/$8.00
10 WATT ZENERS 3.9, 4.7, 5.6, 8.2, 18,22,100,150 or 200V . ea. $ .60
1 WATT ZENERS 4.7, 5.6,10, 12, 15 18 or 22V . . . . . . . . . . . ea. $ .25
MC6860 MODEM CHIP . . . . . . . $9.95

### Silicon Power Rectifiers

| PRV | 1A | 3A | 12A | 50A | 125A | 240A |
|---|---|---|---|---|---|---|
| 100 | .06 | .14 | .30 | .80 | 3.70 | 5.00 |
| 200 | .07 | .20 | .35 | 1.15 | 4.25 | 6.50 |
| 400 | .09 | .25 | .50 | 1.40 | 6.50 | 9.50 |
| 600 | .11 | .30 | .70 | 1.80 | 8.50 | 12.50 |
| 800 | .15 | .35 | .90 | 2.30 | 10.50 | 16.50 |
| 1000 | .20 | .45 | 1.10 | 2.75 | 12.50 | 20.00 |

SAD 1024-a REDICON 1024 stage analog "Bucket Brigade" shift register. $18.95

IN 4148 (IN914) . . . . . . . . . . .15/$1.00

MCA 81 OPTICAL LIMIT SWITCH $1.50

### REGULATORS

309K . . . . . . $ .95
723 . . . . . . $ .50
LM 376 . . . . $ .60
320K-5 or 15V $1.20
320T-5, 15 or 24V . . $ .95
340K-12,15 or 24V. . . . $ .95
340T-5, 6, 8, 12 15,18 or 24V $ .95
78 MG . . . . .$1.35
79 MG . . . . .$1.35

### TRANSISTOR SPECIALS

2N6233-NPN SWITCHING POWER $ 1.95
**MRF-8004 a CB RF Transistor NPN $ 1.50**
2N3772 NPN Si TO-3 . . . . . . . . $ 1.00
2N1546 PNP GE TO-3 . . . . . . . . $ .75
2N4908 PNP Si TO-3 . . . . . . . . $ 1.00
2N6056 NPN Si TO-3 Darlington . $ 1.70
2N5086 PNP Si TO-92 . . . . . . . 4/$ 1.00
2N4898 PNP TO-66 . . . . . . . . . $ .60
2N404 PNP GE TO-5 . . . . . . 5/$ 1.00
2N3919 NPN Si TO-3 RF . . . . . . $ 1.50
MPSA 13 NPN Si TO-92 . . . . . 3/$ 1.00
2N3767 NPN Si TO-66 . . . . . . . $ .70
2N2222 NPN Si TO-18 . . . . . . 5/$ 1.00
2N3055 NPN Si TO-3 . . . . . . . . $ .50
2N3904 NPN Si TO-92 . . . . . . 5/$ 1.00
2N3906 PNP Si TO-92 . . . . . . 5/$ 1.00
2N5296 NPN Si TO-220 . . . . . . $ .50
2N6109 PNP Si TO-220. . . . . . . $ .55
2N3638 PNP Si TO-5 . . . . . . . 5/$ 1.00

### TTL IC SERIES

| | | |
|---|---|---|
| 7400– .13 | 7445– .65 | 74151– .61 |
| 7401– .13 | 7446– .68 | 74153– .61 |
| 7402– .13 | 7447– .58 | 74154– .94 |
| 7403– .13 | 7448– .68 | 74155– .58 |
| 7404– .15 | 7450– .15 | 74157– .55 |
| 7405– .13 | 7472– .25 | 74161– .55 |
| 7406– .16 | 7473– .28 | 74163– .55 |
| 7407– .20 | 7474– .28 | 74164– .85 |
| 7408– .18 | 7475– .45 | 74165– .95 |
| 7409– .18 | 7476– .30 | 74170– 1.68 |
| 7410– .13 | 7480– .31 | 74173– 1.20 |
| 7411– .18 | 7483– .65 | 74174– .95 |
| 7412– .13 | 7485– .87 | 74175– .85 |
| 7413– .36 | 7486– .28 | 74176– .75 |
| 7414– .60 | 7490– .42 | 74177– .75 |
| 7416– .22 | 7491– .58 | 74180– .65 |
| 7417– .25 | 7492– .43 | 74181– 1.90 |
| 7420– .13 | 7493– .43 | 74190– 1.00 |
| 7425– .25 | 7494– .67 | 74191– 1.00 |
| 7426– .22 | 7495– .65 | 74192– .79 |
| 7427– .19 | 7496– .65 | 74193– .79 |
| 7430– .13 | 74107– .28 | 74194– .80 |
| 7432– .22 | 74121– .29 | 74195– .85 |
| 7437– .21 | 74122– .38 | 74196– .86 |
| 7438– .21 | 74123– .45 | 74279– .55 |
| 7440– .13 | 74125– .40 | 74367– .65 |
| 7441– .70 | 74126– .40 | 75491– .50 |
| 7442– .37 | 74150– .94 | 75492– .50 |

1/2 Hr. DATA CASSETTES $1.25
44 Pin Solder Tail .156″ Connector $2.20

MM 5387AA new clock chip which will directly drive LED's 12/24 hrs., 1 supply & alarm $5.95

MCM 6571A 7 x 9 upper & lower case character gen. . . . . . . . . . . . .$10.75

ALCO MINIATURE TOGGLE SWITCHES
MTA 106 SPDT . . . . . . . . . . . $ 1.05
MTA 206 DPDT . . . . . . . . . . . $ 1.70
MTA 206 P-DPDT CENTER OFF $ 1.85
MSD 206 P-DPDT CENTER OFF LEVER SWITCH . . . . . . . . . . $ 1.85

### Full Wave Bridges

| PRV | 2A | 6A | 25A |
|---|---|---|---|
| 100 | | | 1.30 |
| 200 | .75 | 1.25 | 2.00 |
| 400 | .95 | 1.50 | 3.00 |
| 600 | 1.20 | 1.75 | 4.00 |

### SANKEN AUDIO POWER AMPS

Si 1010 G 10 WATTS. . . . . . . . . $ 6.80
Si 1020 G 20 WATTS. . . . . . . . . $13.70
Si 1050 G 50 WATTS. . . . . . . . . $25.90

### TANTULUM CAPACITORS

| | |
|---|---|
| .22UF 35V 5/$1.00 | 6.8UF 35V 4/$1.00 |
| .47UF 35V 5/$1.00 | 10UF 10V $ .25 |
| .68UF 35V 5/$1.00 | 22UF 25V $ .40 |
| 1UF 35V 5/$1.00 | 15UF 35V 3/$1.00 |
| 2.2 UF 20V 5/$1.00 | 30UF 6V 5/$1.00 |
| 3.3UF 35V 4/$1.00 | 47UF 20V $ .35 |
| 4.7UF 15V 5/$1.00 | 68 UF 15V $ .50 |

| 74LS SERIES | | LINEAR CIRCUITS |
|---|---|---|
| 74LS00 – .21 | 74LS126 – .47 | LM 101 – .75 |
| 74LS02 – .21 | 74LS132 – .80 | LM 301/748 .29 |
| 74LS03 – .21 | 74LS136 – .39 | LM307 – .30 |
| 74LS04 – .28 | 74LS138 – .70 | LM 308 – .75 |
| 74LS05 – .28 | 74LS139 – .70 | LM 311 – .75 |
| 74LS08 – .21 | 74LS151 – .70 | LM 318 – 1.20 |
| 74LS09 – .28 | 74LS153 – .70 | LM 319 – .95 |
| 74LS10 – .21 | 74LS155 – .70 | LM 324 – 1.05 |
| 74LS11 – .21 | 74LS156 – .70 | LM 339 – 1.10 |
| 74LS13 – .44 | 74LS157 – .72 | LM 358 – 1.40 |
| 74LS15 – .25 | 74LS160 – .88 | LM 370 – 1.15 |
| 74LS20 – .24 | 74LS161 – .85 | LM 377 – 2.50 |
| 74LS21 – .26 | 74LS162 – .85 | LM 380 – .95 |
| 74LS22 – .26 | 74LS163 – .85 | LM 381 – 1.25 |
| 74LS26 – .32 | 74LS168 – .88 | LM 382 – 1.25 |
| 74LS27 – .32 | 74LS169 – .85 | LM 537 – 2.50 |
| 74LS30 – .26 | 74LS170 – 1.68 | LM 553 – 2.50 |
| 74LS32 – .32 | 74LS173 – 1.10 | LM 555 – .39 |
| 74LS37 – .32 | 74LS174 – 1.00 | LM 556 – .85 |
| 74LS38 – .32 | 74LS175 – .80 | NE540L – 2.25 |
| 74LS40 – .26 | 74LS190 – .95 | 560 – 2.00 |
| 74LS42 – .65 | 74LS191 – .95 | 565 – .95 |
| 74LS47 – .79 | 74LS192 – .95 | 566 – 1.25 |
| 74LS51 – .26 | 74LS193 – .95 | 567 – 1.30 |
| 74LS54 – .26 | 74LS195 – .95 | 703 – .90 |
| 74LS90 – .95 | 74LS196 – .95 | 709 – .25 |
| 74LS74 – .35 | 74LS197 – .95 | 710 – .35 |
| 74LS76 – .35 | 74LS221 – 1.25 | 741C or V – .31 |
| 74LS90 – .55 | 74LS257 – .70 | 747 – .65 |
| 74LS92 – .55 | 74LS258 – .70 | LM 1310 – 2.50 |
| 74LS93 – .55 | 74LS279 – .55 | 1456 – .95 |
| 74LS109 – .38 | 74LS365 – .55 | 1458 – .50 |
| 74LS112 – .38 | 74LS366 – .55 | CA3046 – .75 |
| 74LS113 – .38 | 74LS367 – .55 | 3900 – .49 |
| 74LS114 – .38 | 74LS368 – .55 | 8038CC – 3.90 |
| 74LS125 – .47 | 74LS390 – 1.40 | 791 - Power op amp – 1.95 |

### TRIACS / SCR'S

| PRV | TRIACS 1A | TRIACS 10A | TRIACS 25A | SCR'S 1.5A | SCR'S 6A | SCR'S 35A |
|---|---|---|---|---|---|---|
| 100 | .40 | .70 | 1.30 | .40 | .50 | 1.20 |
| 200 | .70 | 1.10 | 1.75 | .60 | .70 | 1.60 |
| 400 | 1.10 | 1.60 | 2.60 | 1.00 | 1.20 | 2.20 |
| 600 | 1.70 | 2.30 | 3.60 | | 1.50 | 3.00 |

**Terms: FOB Cambridge, Mass. Send Check or Money Order. Include Postage, Minimum Order $5.00, COD'S $20.00**

Send 25¢ for our catalog featuring Transistors and Rectifiers
145 Hampshire St., Cambridge, Mass.

**SOLID STATE SALES**
**P.O. BOX 74K**
SOMERVILLE, MASS. 02143 TEL. (617) 547-7053

**WE SHIP OVER 95% OF OUR ORDERS THE DAY WE RECEIVE THEM**

# ADVANCED COMPUTER PRODUCTS

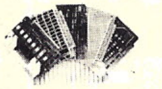
A38

THE FIRST TO OFFER PRIME PRODUCTS TO THE HOBBYIST AT FAIR PRICES NOW LOWERS PRICES EVEN FURTHER!

**1. Proven Quality** Factory tested products only, no re-tests or fallouts. Guaranteed money back. We stand behind our products.

**2. Same Day Shipment** All prepaid orders with cashiers check, money order or charge card will be shipped same day as received.

## MOTOROLA 6800 COMPATIBLE MODULES



| | |
|---|---|
| MEK 6800 D2 KIT | $235.00 |
| 9600 6800 MPU | 595.00 |
| 9601 16 Slot Mother Bd. | 175.00 |
| 9602 16 Slot Card Cage | 75.00 |
| 9603 8 Slot Mother Bd. | 100.00 |
| 9610 Proto Board | 36.00 |
| 9615 4K EPROM Module (1702A) | 250.00 |
| 9620 16 Port Parallel I/O | 375.00 |
| 9626 8K Static RAM Module | 295.00 |
| 9626K 8K Static RAM Kit | 225.00 |
| 9630 Extender Card | 60.00 |
| 9650 8 Port Duplex Asyn Serial I/O | 395.00 |
| Connectors $6.50 | 6/84.00 |

All assembled & tested not Kits

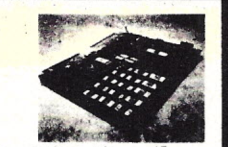
PLUS MOTOROLA TV MONITORS-PRIME

Model M3560-155 L01 12" display 219.95
Model M2000-155 9" display 199.95

Add $10.00 for shipping

## AMI 6800 KIT EVK 99

only $133.00



Send for complete details

| | |
|---|---|
| EVK 99 Kit | $133.00 |
| EVK 100 Kit | 269.95 |
| EVK 200 Kit | 449.95 |
| EVK 300 Assembled | 699.00 |
| Universal Kluge Board | 95.00 |
| 16K Byte RAM Board | 75.00 |
| 6 Slot Motherboard | 35.00 |
| Extender Board | 45.00 |
| Video Board (avail. future) | 95.00 |
| Connectors . . . $6.50 | 12/59.95 |
| Solid Frame Chassis | 120.00 |
| Frame Chassis | 69.95 |
| Micro Assembler ROM | 30.00 |
| Proto ROM | 30.00 |
| Tiny Basic Papertape | 20.00 |
| Tiny Basic EPROM | 125.00 |

Hardware or Programming Manuals 15.00 ea.

## LOGOS 8K STATIC MEMORY

$125.99 Kit (450ns.)

| | |
|---|---|
| Assembled & tested | $149.95 |
| 250 ns. Kit | 149.95 |
| Assembled & tested | 179.95 |

Features: Lowpower, Dip Switch Selectable memory protect down to 256 Bytes, addressing on 1K Boundaries, No wait states, fully buffered, battery back-up.

NOW 1 Year Field Experience

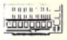
SPECIAL OFFER:
Buy 4 Units Only $117.00 ea.

## 8080A CPU KIT
$99.95 Kit

Assembled & tested $129.95

FEATURES:
S-100 bus compatible, complete CPU with eight level vector interrupt. Includes sockets.
Add $5.00 for 8080 Manual

## BYTEUSER 8K EPROM $ 64.95 Kit



| | |
|---|---|
| Assembled & tested | $ 94.95 |
| Bare PC Board w/Data | 21.95 |
| 8K EPROM (8) 2708 | $112.00 |

Features: Power on Jump, Reset Jump, all socketed with top quality PC Board material.

SPECIAL OFFER:
Buy 4 Units Only $59.95 ea.

## KEYBOARDS

73 Keys with 2 position Thumbwheel, full ASCII II, dual mode. Special purchase from Xerox manufactured by Microswitch. Only 50 units available.

ONLY $59.95

## TARBELL FLOPPY INTERFACE

Complete Kit Only $179.95
Assembled & tested $269.95

FEATURES: S100/Altair/IMSAI compatible, compatible to most disc drives including Persci, Innovex, GSI, Shugart and others.

SOFTWARE: Uses CPM which is available for $70.00
CPM documentation add $20.00

Same Day Shipment

## Z-80 CPU KIT $169.95 Kit (2mhz)

Low Price
(regular price 269.95)

Assembled & tested $199.95

FEATURES:
S-100 IMSAI/Altair compatible, completely compatible to TDL hardware and software. Can be used at 4MHz with Z-80A.
Add $7.00 for sockets.
Add $5.00 for Z80 manual.

NOW IN STOCK FOR IMMEDIATE SHIPMENT.

## NAKED PC BOARD SALE S-100

| | |
|---|---|
| Z-80 CPU | $31.95 |
| 2708 EPROM 8K | 21.95 |
| 8K Static RAM | 21.95 |
| 32K Static RAM | 59.95 |
| Floppy I/O | $39.95 |
| Cassette I/O | 29.95 |
| Proto Bd. | 27.95 |
| Extender | 15.95 |

## DATA BOOKS

| | |
|---|---|
| NSC Digital | $ 3.95 |
| NSC Linear | 4.95 |
| NSC Linear A/N Vol. I | 2.95 |
| NSC Linear A/N Vol. II | 2.95 |
| NSC CMOS | 2.95 |
| NSC Audio | 2.95 |
| NSC Voltage Reg | 2.95 |
| NSC Memory | 3.95 |
| Intel Data Book | 3.95 |
| NSC Special Functions | 3.25 |
| NSC Mos/LSI | 4.25 |
| NSC Transistors | 2.95 |
| Iasis Microprocessor Handbook | 7.95 |
| Iasis Complete up Learning Course | 99.50 |
| Intel MCS80 Manual | 4.95 |
| Intel MCS40 Manual | 4.95 |
| AMI MOS Catalog | 2.95 |
| Raytheon Linear | 2.95 |
| Raytheon Quads/Duals | 1.95 |
| New GI mos | 4.95 |
| Osborne Introto Microputer Vol. 0 | 7.50 |
| Osborne Introto Microputer Vol. I | 7.50 |
| Osborne Introto Microputers Vol. II | 15.00 |
| Osborne 8080 Programming for Logic Design | 7.50 |
| Osborne 6800 Programming | 7.50 |

## UV "EPROM" ERASER

Model UVS-11E only $59.95

Holds (4) chips at a time. Special holding tray with UVA Absorber. Exclusive safety interlock system.

Backed by 45 years UV experience.

## WOW 1771 FLOPPY SPECIAL

While they last $32.95
regular price $55.95

Western digital P/N 1771A with App. Note & Data

Only 100 units available

## MICROPROCESSORS

| | |
|---|---|
| ● Z-80 | $21.95 |
| Z-80A | 29.95 |
| F-8 (3850) | 19.95 |
| 2650 | 24.95 |
| CD1802 | 19.95 |
| 8080A | 12.95 |
| 8085 | 29.95 |
| 8008-1 | 14.95 |
| 2901 | 21.95 |
| 2901A | 29.95 |
| TMS 9900JL | 89.95 |
| CP1600 | 39.95 |
| 6502 | 19.95 |
| IM6100 | 29.95 |
| ● 6800P | 19.95 |
| 6802P | 32.95 |

## SUPPORT DEVICES

| | |
|---|---|
| ● 3881 (Z80 PIO) | $12.95 |
| ● 3882 (Z80 CTC) | 12.95 |
| 3851 (F8 Program) | 14.95 |
| 3853 (F8 Memory I/O) | 14.95 |
| 8212 8 bit I/O | 3.50 |
| 8214 Priority Interrupt | 9.95 |
| 8216 Bus driver | 3.75 |
| 8224 Clock Gen. | 4.85 |
| 8224-4 4MHz CLK Gen. | 12.95 |
| ● 8T26 Busdriver | 2.25 |
| 8226 Busdriver | 3.95 |
| 8228 Sys Controller | 7.95 |
| 8238 Sys Controller | 9.95 |
| 8251/9551 Prog. Comm. I/O | 11.95 |
| 8253 Interval Timer | 21.95 |
| 8255/9555 Prog. Periph. I/O | 11.95 |
| 8257 DMA Control | 32.95 |
| 8259 Prog. Interrupt | 32.95 |
| 6810-1 128 x 8 RAM | 4.95 |
| ● 6820 PIA | 7.95 |
| 6834-1 EPROM 4K | 14.95 |
| 6834 EPROM 4K | 16.95 |
| 6850 ACIA | 9.95 |
| 6852 Serial Adapter | 11.95 |
| 6860 Modem | 14.95 |
| 6862 RPS Modulator | 17.95 |
| ● 6871B 1.0 MHz Clock OSC | 26.95 |
| 6880 MPU Busdriver | 2.95 |
| 68 MinII Minibug 2 | 49.95 |
| 68 MinIII Minibug 3 | 74.95 |
| 6831/10224 Micro Assembler Rom | 30.00 |
| 6831/11003 Proto Rom | 30.00 |
| 1821SCD 1K RAM | 15.95 |
| 1822SCD 256 x 4 RAM | 16.95 |
| 1824CD 32 x 8 RAM | 9.95 |
| 1852CD 8 bit I/O | 10.95 |
| 1856CD I/O | 8.95 |
| 1857CD I/O | 8.95 |

## PROM'S

| | |
|---|---|
| ● 2708 | $14.95 |
| 2708S 650ms. | 9.75 |
| 1702A | 7.95 |
| ● 1702-6 | 3.50 |
| 5204 | 16.95 |
| ● IM5610 | 2.95 |
| 6834 | 16.95 |
| 6834-1 | 14.95 |
| 82S123 | 2.95 |
| 82S126 | 4.50 |
| 82S129 | 2.95 |
| 8223 | 2.95 |
| 5203 | 6.95 |

Programming Available

## RAMS DYNAMIC

| | |
|---|---|
| ● 4115 | $18.95 |
| 4116 16K | 34.95 |
| TMS4050 | 4.95 |
| TMS4060 22 pin | 4.25 |
| WD4060 22 pin | 3.25 |
| 4096 16 pin | 4.95 |
| 2104 | 4.95 |
| 2107 | 3.95 |
| ● 2107B-4 | 4.25 |
| MM5261 | 3.95 |
| MM5262 | 1.95 |
| MM5270 | 4.95 |
| MM5280 | 4.95 |
| 1103 | 1.95 |
| 4008 L | 2.95 |

## FLOPPY DISK

| | |
|---|---|
| WD 1771 Floppy | 32.95 |
| WD 1771 B01 | 52.95 |

## CHARACTER GENERATORS

| | |
|---|---|
| R032513 Upper 5V | $10.95 |
| R032513 Lower 5V | 9.95 |
| HD0165 | 9.95 |
| MC6571A | 10.80 |
| MC6574 | 14.50 |
| MC6575 | 14.50 |

## KEYBOARD ENCODERS

| | |
|---|---|
| AY52376 | $14.95 |
| AY53600 | 14.95 |

## UARTS/USRTS

| | |
|---|---|
| AY51013A (5V) | $ 5.50 |
| TR1602B (5V) | 4.50 |
| AY51014 (5-14V) | 8.95 |
| AY51015 (5V) | 9.95 |
| IM6402 (5V) | 12.95 |
| IM6403 (5V) | 12.95 |
| S2350 | 10.95 |
| WD1671B Astros | 19.95 |

## BAUD RATE GENERATORS

| | |
|---|---|
| MC14411 | $11.95 |
| WD1941 Dual | 9.95 |
| 34702 | 13.95 |

## TV CHIPS

| | |
|---|---|
| ● MM5320 TV Synch | $ 9.95 |
| MM5369 Prescaler | 3.95 |
| MM57100 Game Chip | 16.95 |
| MM57104 Clock | 3.75 |
| LM1889 Modulator | 3.95 |
| CW300 Saw Function | 1.50 |
| AY38500-1 TV Game | 9.95 |
| AY38600-1 Color TV Game | 24.95 |
| AY38615-1 Color Converter | 8.95 |
| ● AY38700-1 Tank Chip | 29.95 |
| RF Modulator | 8.95 |

## ORGAN CHIPS

MM5554, MM5555, MM5556 (1 each) . . . NSC Set $19.95

## CLOCK CHIPS

| | |
|---|---|
| MM5314 | $ 4.95 |
| MM5316 | 4.95 |
| MM5375 | 4.95 |

## STATIC RAM HEADQUARTERS

We maintain largest inventory of Static Rams in the West.

| | 1-24 | 25-99 | 100 |
|---|---|---|---|
| ● 21L02 (450 ns.) | 1.30 | 1.25 | 1.15 |
| 21L02 (250 ns.) | 1.95 | 1.80 | 1.50 |
| 2102 | 1.25 | 1.15 | 1.10 |
| 2102-1 | 1.30 | 1.25 | 1.15 |
| 91L02APC | 1.75 | 1.65 | 1.50 |
| 21L11-1 | 4.25 | 4.10 | 3.95 |
| 2111-1 | 4.10 | 3.95 | 3.85 |
| 2112-1 | 3.00 | 2.80 | 3.69 |
| 21L01-1 | 2.95 | 2.85 | 2.75 |
| 2101-1 | 2.95 | 2.85 | 2.75 |
| 2114 (650 ns.) | 12.95 | 12.25 | 11.45 |
| 2114 (450 ns.) | 14.95 | 14.25 | 13.45 |
| 4200A | 12.95 | 12.25 | 11.45 |
| ● upD410 (4200) | 10.95 | 10.50 | 9.95 |
| 4402 | 12.95 | 12.25 | 11.45 |
| 5101C-E | 11.95 | 11 25 | 10.25 |
| 74C89 | 3.25 | 3.05 | 2.85 |
| 7489 | 2.25 | 1.99 | 1.88 |
| 8599 | 1.88 | 1.75 | 1.60 |
| 74S189 | 4.50 | 4.30 | 4.25 |
| 74S200 | 4.50 | 4.30 | 4.25 |
| 74S201 | 4.50 | 4.30 | 4.25 |
| 1101A | 1.49 | 1.29 | 1.10 |
| 3107 | 3.95 | 3.70 | 3.25 |

## A TO D CONVERTORS

| | |
|---|---|
| 8700CJ | $13.95 |
| 8701CN | 21.95 |
| 8750CJ | 13.95 |
| 1408L8 | 5.95 |
| ● 148L6 | 3.95 |

## DISPLAYS/OPTO

| | |
|---|---|
| DL 704/707 CC/CA .300 | 1.25 |
| FND359 CC .357 | .95 |
| FND 500/507 CC/CA .500 | 1.35 |
| FND 503/510 CC/CA .500 | .95 |
| FND 800/807 CC/CA .800 | 2.50 |
| Bowmar 9 digit bubble | .99 |
| FSC 8024 4 digit CC .800 | 4.95 |
| HP7340 HEX Display | 15.95 |
| TIL 305 5 x 7 Array | 4.50 |
| TIL 306 7 seg w/logic | 8.95 |
| TIL 308 7 seg w/logic | 8.95 |
| TIL 309 7 seg w/logic | 7.95 |
| TIL 311 HEX Display | 9.25 |
| MA1003 12 auto clock | 17.95 |
| MA1002 4 digit clock module | 9.95 |
| MA1010 4 digit clock module | 9.95 |
| NSN 373/374 dual CC/CA .300 | 2.20 |
| NSN 583/584 dual CC/CA .500 | 2.60 |
| NSN 783/784 dual CC/CA .700 | 3.00 |
| 4N25 Opto Isolater | 3/1.99 |
| MCT 2 Opto Isolater | .89 |
| 4N33 Darlington ISOL | 1.75 |
| Red Led's .185 Dia. | 5/1.00 |
| Green/Yellow | 4/1.00 |

## SOCKETS

| | | | |
|---|---|---|---|
| 8 Pin ww | .32 | 8 Pin S/T | .17 |
| 14 Pin ww | .37 | 14 Pin S/T | .20 |
| 16 Pin ww | .38 | 16 Pin S/T | .22 |
| 18 Pin ww | .60 | 18 Pin S/T | .31 |
| 20 ww | .90 | 22 Pin S/T | .35 |
| 22 Pin ww | .93 | 24 Pin S/T | .41 |
| 24 Pin ww | .93 | 28 Pin S/T | .49 |
| 40 Pin ww | 1.45 | 40 Pin S/T | .63 |

## CONNECTORS

| | |
|---|---|
| 6 Pin Single S/E | 1.49 |
| 15/30 Dual S/E | 1.95 |
| 18/36 Dual S/E | 2.35 |
| 22/44 Dual S/E | 2.95 |
| 43/86 Dual S/T | 6.50 |
| 43/86 Dual W/W | 6.50 |
| 50/100 IMSAI W/W | 4.75 |
| 50/100 IMSAI S/T | 4.75 |
| 50/100 Altair W/W | 5.95 |
| 50/100 Actair S/T | 5.95 |

25 Pin-D Subminature

| | |
|---|---|
| DB25P | 3.25 |
| DB25S | 3.75 |
| Hood | 1.25 |

IMSAI Card Guides . . 4/1.00

RS232 CONNECTOR SET W/Hood $6.50

## CRYSTALS

| | | | |
|---|---|---|---|
| $1.0MH_3$ | $ 5.85 | $10.0MH_3$ | 4.95 |
| 2.0 | 5.85 | 13.0 | 4.95 |
| 2.097152 | 5.85 | 14.31818 | 4.95 |
| 2.4576 | 5.85 | 18.0 | 4.95 |
| 3.579545 | 3.95 | 18.432 | 5.95 |
| 4.0 | 4.95 | 20.0 | 4.95 |
| 4.194304 | 5.95 | 22.1184 | 5.95 |
| 4.91520 | 5.95 | 27.000 | 5.95 |
| 5.0 | 4.95 | 36.000 | 5.95 |
| 5.7143 | 5.95 | 48.000 | 5.95 |
| 6.0 | 4.95 | 100KC | 12.95 |

## NEW CTS DIPSWITCHES

| | | | |
|---|---|---|---|
| CTS206-4 | $1.75 | CTS208-8 | $1.95 |
| CTS206-5 | $1.75 | CTS209-9 | $1.95 |
| CTS206-6 | $1.75 | CTS209-10 | $1.95 |
| CTS207-7 | $1.75 | | |

We also stock full of 7400, 74LS, 74L Linear and CMOS Send for pricing or use our competitors STD catalog pricing.

NEW 1978 CATALOGUE AVAILABLE FEB. SEND .25 POSTAGE

## 74LS00 TTL

| | | | |
|---|---|---|---|
| 74 LS00 | .22 | LS86 | .55 |
| LS02 | .24 | LS90 | 1.10 |
| LS04 | .24 | LS92 | 1.10 |
| LS05 | .28 | LS93 | 1.10 |
| LS08 | .28 | LS95 | 1.69 |
| LS10 | .28 | LS107 | .56 |
| LS13 | .79 | LS109 | .56 |
| LS14 | 1.69 | LS112 | .56 |
| LS20 | .26 | LS113 | .56 |
| LS27 | .38 | LS123 | .99 |
| LS30 | .26 | LS132 | 1.10 |
| LS32 | .34 | LS136 | .99 |
| LS40 | .26 | LS138 | 1.39 |
| LS42 | 1.10 | LS139 | 1.49 |
| LS51 | .26 | LS151 | 1.49 |
| LS73 | .56 | LS153 | 1.29 |
| LS74 | .56 | LS155 | 1.49 |
| LS75 | .79 | LS157 | 1.29 |
| LS76 | .59 | LS158 | 1.29 |
| LS83 | 1.69 | LS161 | 1.39 |
| LS85 | 1.99 | LS162 | 1.39 |

| | | | |
|---|---|---|---|
| LS163 | 1.39 | LS367 | .99 |
| LS164 | 1.39 | LS368 | .99 |
| LS174 | 1.39 | LS377 | 1.29 |
| LS175 | 1.39 | 74200 | 5.95 |
| LS190 | 2.49 | 74251 | 1.69 |
| LS191 | 2.49 | 74279 | 1.10 |
| LS192 | 2.49 | 74298 | 1.99 |
| LS193 | 2.49 | 74365 | .79 |
| LS194 | 2.49 | 74366 | .79 |
| LS195 | .52 | 74367 | .79 |
| LS251 | 1.39 | 74368 | .79 |
| LS253 | 1.49 | LS378 | 1.29 |
| LS257 | 1.29 | 81LS95 | 1.10 |
| LS258 | 1.29 | 81LS96 | 1.10 |
| LS279 | 1.10 | 81LS97 | 1.10 |
| LS283 | 1.10 | 81LS98 | 1.10 |

## MONTHLY SPECIALS

| | |
|---|---|
| up D371 Mag Tape Control | $49.95 |
| AY5-3550 4-3/4 Digit DVM | 24.95 |
| MK5007/5009 Counter | 6.95 |
| AY5-3507 DVM Chip | 12.95 |
| ICM7208IPI Ctr./Disp/Driv. | 16.95 |
| ICM7209 IPI Ctr/Driver | 19.95 |
| ICM7045 IPI Stop Watch | 18.95 |
| G115M 6 Channel MOS FET SW. | 1.50 |
| 4116/416 16K Dynamic RAM | 34.95 |
| Intel 3404 6 bit Latch | 2.95 |
| 1488/1489 RS232 Driver | 1.25 |
| 8T97 Buffer | 1.25 |
| 8130/8131 | 2.45 |
| 8833/8835 | 1.99 |
| 74367/74368 | 6/5.00 |
| 75451/52/53 | 10/2.50 |
| 8T26 | 2/4.50 |
| 78L05 | 2/1.00 |
| LM318H | 1.00 |

**4N25 Opto/Iso .69**

SPECIAL 7/$1.00

7405 MPS6530
7416 MPS3568
7438 MPS6516
7440 MPS6522

SPECIAL 2/$1.00

7442 74123
7490 74145
7493 74153
7495 74LS195

| | | | |
|---|---|---|---|
| MCT2 | $ .89 | MPS3642 | 5/1.00 |
| 2N3638A | 5/1.00 | MPS6571 | 5/1.00 |
| 2N3640 | 5/1.00 | MPS3693 | 5/1.00 |
| 2N2369A | 5/1.00 | MPS6530 | 5/1.00 |
| 2N3692 | 5/1.00 | MPS3646 | 5/1.00 |
| MSPS4318 | 5/1.00 | 7525J | .50 |
| MPS5401 | 5/1.00 | 7524J | .60 |
| MPS6516 | 5/1.00 | 74S258 | .60 |
| MPS6544 | 5/1.00 | 7489 | 1.49 |
| MPS3568 | 5/1.00 | 74145 | .60 |
| MPS6522 | 5/1.00 | 74H55 | .19 |
| MPS404A | 5/1.00 | 715DC | 1.49 |
| 2N5462 | 5/1.00 | 74199 | 1.00 |

## COMP KITS & SYSTEMS:

| | |
|---|---|
| MEK6800D2 Kit | $235.00 |
| KIM1 6502 | 245.00 |
| EVK99 6800 Kit | 133.00 |
| Technico 9900 Kit | 299.00 |
| Intercept Jr. 6100 Kit | 281.00 |
| Iasis 8080 Computer Book | 499.00 |
| NSC SC/MP Kit | 99.00 |
| NSC Keyboard Kit | 95.00 |
| Low Cost S-100-BUS with 8 slots & Power Supply | 159.95 |
| 8K Ram Kit (Logos I) | 125.95 |
| Byteuser 8K Eprom Kit | 64.95 |
| ZPU Kit (Z80) | 295.00 |
| TDL ZPU Kit (Z80) | 269.00 |
| Tarbell cassette I/O | 115.00 |
| S100 8 slot Motherboard with connectors (expandable) | 69.95 |
| S100 Extender Board | 15.95 |
| S100 Proto Board | 27.95 |
| Vector 8800 Proto Board | 19.95 |
| IMSAI 8080 w/22 slot | 699.00 |
| Cromemco Z-2 | 595.00 |
| TDL XITAN alpha 1 | 769.00 |
| Computalker | 385.00 |
| Heuristics Speechlab | 299.00 |
| SOROC IQ120 Terminal | 995.00 |
| 8K Ram Board (Logos) | 21.95 |
| 8K Eprom Board | 21.95 |
| S100 32K Ram Kit | 875.00 |
| Bytesaver Kit | 145.00 |

## NOTICE

We are looking for software support packages to offer in our new catalogue. Please write or call if you want to participate.

All Shipments FCM or UPS. Orders under $100.00 add 5% handling and postage. Orders over $100.00 add 2.5% handling & postage. Mastercharge/Bankamericard/COD accepted w/25% deposit. California Residents add 6% tax. Foreign Orders add 8% handling. All parts prime factory tested guaranteed. Same day shipment. Add 25 cents for Data.

Retail pricing may vary from Mail Order Pricing.

P. O. BOX 17329 Irvine, California 92713 New Phone (714) 558-8813
TWX: 910-595-1565

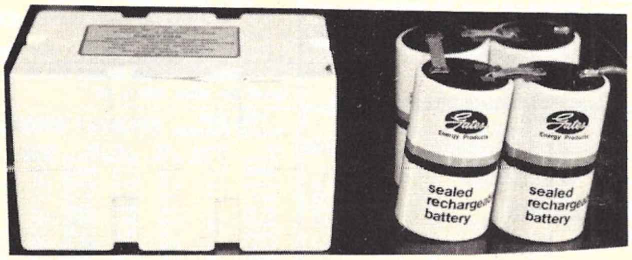
Retail Store Open Mon. – Sat.
Located at 1310 "B" E. Edinger,
Santa Ana, CA 92705

## SERIES SN7400 BENT PIN SPECIAL

We have a large lot of SERIES SN7400 ICs, plus a few linears, which were removed from wire-wrap boards. Included are just about every SN7400 in the book, plus op-amps, phase lock loops, and others found in a sophisticated computer system. The only defect we have been able to find is bent pins, when these ICs were removed from the wire-wrap board. We are selling these ICs in lots of 125, which will contain a good cross section of the mixture. Money back guarantee if not 100 per cent satisfied that you are getting many times your moneys worth. We cannot allow any choice at this price, but you are bound to get a good working supply of these useful devices.

| | | | |
|---|---|---|---|
| STOCK No. 4591K | 125 piece SN7400 and linear kit, | 10.00 | 3/25.00 |

## GATES ENERGY CELLS




These energy cells were used to make up batteries for back up supplies for computers, in case of power outage. We have 4 different batteries, 6 volts 10 amp hours, 8 volts 10 amp hours, 12 volts 2.5 amp hours and 12 volts 5 amp hours. The specs on these batteries show them to be good for up to 2000 recharge cycles. These high capacity batteries are useful for a multitude of things other than computers. They come either in a plastic case, or individual cells tied together.

We provide application data, and charging information and circuits for these batteries.

| | | | |
|---|---|---|---|
| STOCK No. 5572K | 6 volt 10 amp. hour battery | 22.50 | 2/40.00 |
| STOCK No. 5573K | 8 volt 5 amp. hour battery | 19.50 | 2/37.00 |
| STOCK No. 5574K | 12 volt 2.5 amp. hour battery | 12.50 | 2/23.00 |
| STOCK No. 5575K | 12 volt 5 amp. hour battery | 22.50 | 2/40.00 |

## 3 NEW BLOCKBUSTER TRANSFORMERS

TRANSFORMER 1. 115 primary, Secondary 1, 30 V @2A. Secondary 2, 16.5 V @1.2A, Secondary 3, 16 V @ 3.5A, Secondary 4, 9.5 V @3.5A. STOCK NO. 6667K 10 lbs. $10.95 2/$20.00

TRANSFORMER 2. Dual primary. Secondary 1, 12V @5A. Secondary 2, 24 V @9A. Secondary 3, 14 V @20A. Secondary 4, 125V @1.5A. Wt. 16 lbs. STOCK No. 6675N $18.95 ea. 2/$36.00

TRANSFORMER 3, 2 different primaries. Following 3 voltages & currents with primary 1. Secondary 1, 9.8 V @8.8A. Secondary 2.18 volts @5A. Secondary 3, 21 V @6.5A. Primary 2, secondary 1, 5.8 V @8.8A, Secondary 2, 10 V @5A.
Secondary 3, 12 V @6.5A. Wt. 10 lbs. STOCK No. 6675K $12.95 2/$24.00

## WIRE WRAP PROTOTYPE BOARDS

Wire Wrap is the thing today, whether you are adding memory to your computer, building from scratch, designing new circuits, etc. We have 4 boards, 2 out of equipment, that have wire on them that must be removed (easy with an OK wire-wrap tool), and 2 virgin boards. Board 6558K has from 75 to 100 sockets 14 & 16 pin. Board 6559K has from 40 to 50 sockets, 14 & 16 pin. Board 6592K has 40 16-pin sockets, and 4 LSI sockets, and is gold-plated. All pins are brought up to the top of this board for ease in wiring. Board 5561K has 87½ sockets, 28 16-pin sockets and a 4-pin socket. Has heavily by-passed Vcc and ground planes.

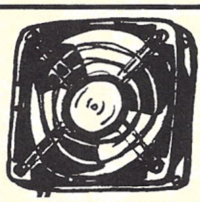
| | | | | |
|---|---|---|---|---|
| STOCK No. 6558K | 75 to 100 sockets | 5¾x13¾ | $18.75 ea. | 2/$36.00 |
| STOCK No. 6559K | 40 to 50 sockets | 6"x6½" | 11.75 ea. | 2/$22.00 |
| STOCK No. 6592K | 40 16-pin sockets 4 LSI sockets | 6¾x8¾" | $24.50 ea. | 2/$45.00 |
| STOCK No. 5561K | 88½ sockets | 4½"x14½" | $29.50 ea. | 2/$55.00 |

## VIDEOCUBE

THE COMPUTER / TV INTERFACE

VIDEOCUBE is a self-contained oscillator and modulator, which allows easy interface with any device having a video output, and a standard TV set. When properly used, the output of your video camera, video game or video output of your computer is displayed on channel 3. Easy switching from TV to monitor. The VIDEOCUBE was completely described in August issue of RADIO-ELECTRONICS. We supply a reprint of this article. Has FCC approval for radiation.

| | | | |
|---|---|---|---|
| STOCK No. 5500K | Complete kit of parts with data. | $13.95 | 2/26.00 |



## ROTRON WHISPER FANS

KEEP YOUR EQUIPMENT COOL, with ROTRON WHISPER FANS. 115 VAC. 7 Watts. These fans have been removed from equipment, and are fully guaranteed. 4½"x4½"x1½"

STOCK No. 5520K $6.95 ea. 2/12.00

MAIL ORDER

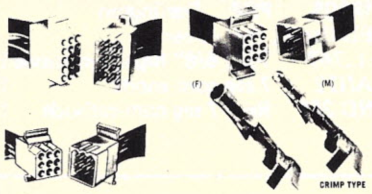
**DELTA ELECTRONICS**
**P.O. BOX 2, 7 Oakland St.**
**Amesbury, Mass. 01913**

Over the counter salesrooms

**ATLANTA GA**
DELTA ELECTRONIC HOBBIES
5151 Buford Hwy.
Doraville, Atlanta, Ga.

**BOSTON MASS.**
DELTA ELECTRONICS
WAREHOUSE OUTLET
590 Commonwealth Ave.
Boston, Mass.

D13

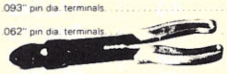



## NYLON CONNECTORS

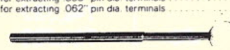
Mfg. by Molex

| Complete Connectors Per Pkg. | Type No. | Class | Description | Price Ea. Pkg. |
|---|---|---|---|---|
| 5 | 1625-1PRT | Min.(.062') | 1 Circuit | $1.75 |
| 3 | 1625-2PRT | " | 2 Circuit | 1.90 |
| 3 | 1625-3PRT | " | 3 Circuit | 2.10 |
| 2 | 1625-4PRT | " | 4 Circuit | 2.10 |
| 2 | 1625-5PRT | " | 5 Circuit | 2.20 |
| 2 | 1625-6PRT | " | 6 Circuit | 2.35 |
| 1 | 1649-8PRT | " | 8 Circuit | 1.55 |
| 1 | 1625-9PRT | " | 9 Circuit | 1.75 |
| 1 | 1625-12PRT | " | 12 Circuit | 1.90 |
| 1 | 1625-15PRT | " | 15 Circuit | 2.30 |
| 1 | 1625-24PRT | " | 24 Circuit | 3.25 |
| 1 | 1772-36PRT | " | 36 Circuit | 4.55 |
| 5 | 1619PRT | Std.(.093") | 1 Circuit | 1.75 |
| 3 | 1545PRT | " | 2 Circuit | 1.90 |
| 3 | 1396PRT | " | 3 Circuit | 2.10 |
| 2 | 1490PRT | " | 4 Circuit | 2.10 |
| 2 | 1653PRT | " | 5 Circuit | 2.20 |
| 2 | 1261PRT | " | 6 Circuit | 2.35 |
| 1 | 1292PRT | " | 9 Circuit | 1.80 |
| 1 | 1360PRT | " | 12 Circuit | 1.90 |
| 1 | 1375PRT | " | 15 Circuit | 2.45 |

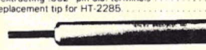
**Prototype hand tools** combine efficiency with economy. Ideal for prototype or limited production runs.
HT-1919 for .093" pin dia. terminals .......... $8.95 each
HT-1921 for .062" pin dia. terminals .......... $8.95 each



**Econo-Extractor** removes terminal from nylon connector housing with smoothness and ease.
HT-2054 for extracting .093" pin dia. terminals .......... $2.25 each
HT-2023 for extracting .062" pin dia. terminals .......... $2.25 each



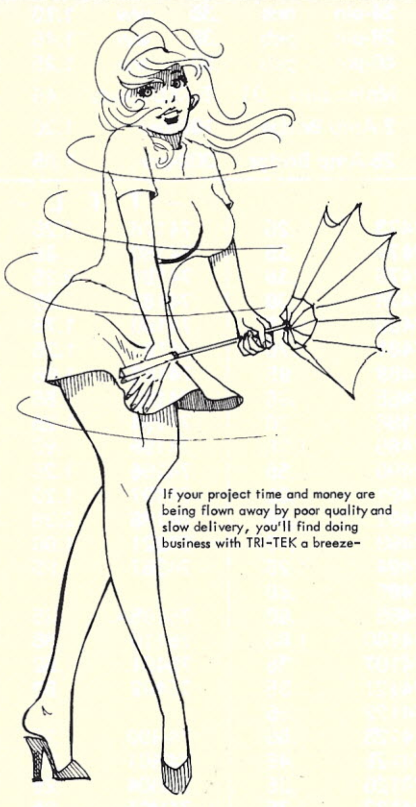
**Deluxe ejector tools**, spring-loaded for simple, efficient removal of terminal from nylon connector housing-extracts either male or female terminals of same pin diameter.
HT-2038 for extracting .093" pin dia. terminals .......... $6.70 each
HT-1010-282 Replacement tip for HT-2038 .......... $2.50 each
HT-2285 for extracting .062" pin dia. terminals .......... $6.70 each
HT-1672-3 replacement tip for HT-2285 .......... $2.50 each



### INTEGRATED CIRCUITS

| | |
|---|---|
| 555 Timer 8 pin mini-DIP | .49 |
| 741 Compensated OP-Amp 8 pin DIP | .37 |
| LM 1889N RF Video Modulator | 7.45 |
| CA3130 Bipolar/Mos-FET Op Amp | 1.19 |
| CA3140 MOS-FET Op Amp, Bi-polar out | .99 |
| LM3909 Lo Voltage Led Pulser | 1.50 |
| LM3911 Temp Control CHIP | 1.50 |
| Signetics 2504TA 1024 bit S.R. memory (1404A) | $1.75 |
| MCM 6571P Character Generator | 9.95 |
| MCM6571AP Character Generator | 9.95 |
| MCM6575P Character generator .............. | $14.88 |
| 50240 Top octave generator .............. | $14.95 |
| LH0070-1H Precision (.3%) 10V Reference Amp | 5.35 |
| LHOO70-2H Extra Precise (.05%) 10V Reference Amp | 10.55 |
| LM399H Temp Stabilized Zener | 5.95 |
| AF100-1CJ Active Filter, State Variable | 7.50 |
| LM2907N Tachometer F/V Converter | 2.65 |
| LM1812N Ultra-sonic Transceiver | 9.15 |
| LM1815 Adaptive Sense Amp for Tachometer | 5.73 |
| TL170 TO-92 Hall effect switch w/spec sheets | 1.25 |
| MC14409P Telephone Rotary Pulser | 10.98 |
| MC14419P Touch Pad Converter for 14409 | 4.25 |
| MC14411P Baud Rate Generator | 11.98 |
| MC14412VP CMOS Modem Chip | 16.95 |
| MM57109N Number Cruncher Micro | 18.95 |
| 74C915 7 Segment to BCD Converter | 2.99 |
| 74C922 16 Key Keyboard Encoder | 6.35 |
| 74C923 20 key Keyboard Encoder | 6.45 |
| 74C925 4 Decade Counter w/latches | 12.00 |
| 74C926 4 Decade Counter w/carry | 12.00 |
| 74C935-1 3½ Digit DVM CMOS Chip | 16.98 |
| 9601 Retriggerable One shot .............. | .50 |
| MC4015P Hi Speed quad "D" low power TTL .... | $1.00 |

New UHF Prescaler IC. Nationals' DS8629 is a fixed ratio, divide by 100 counter combining ECL and lo-power Schottky. Single end or differential mode. Operates from D.C. to typically 160 MHz (135 Guaranteed minimum) TTL compatible, single supply, positive or negative edge trigger. 100mV to 1V input.
DS8629N .................................. $6.14
Specs...................................... .40





ZIF-14 $3.20
ZIF-16 3.49
ZIF-18 3.95
ZIF-22 4.50
ZIF-24 4.85
ZIF-40 7.19
(Last two digits of part number show number of pins per socket)

## *Unique Features of the Welcon Zero Insertion Force Dual-In-Line Sockets . . .*

- **Capability to accept devices with up to .070" lead spread variation.**
- **Accepts lead lengths as short as .075".**
- **Sturdy terminals that fit .025" diameter printed circuit board holes.**
- **"Zero" insertion, positive wiping contacts.**
- **"ZIF" eliminates costly damage to fragile LSI devices**

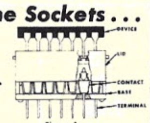


Figure 1
Device before insertion.

Why struggle to insert DIP's into sockets when you can do it the easy way . . . the Welcon way . . . with a new design concept that provides "zero insertion force" and positive wiping.

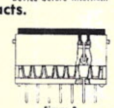


Figure 2

The "604" series of sockets, with all of its positive features and higher reliability, is small in size. High density packaging can be achieved for burn-in and production.

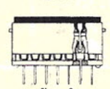


Figure 3

**OPERATION:** Lid is moved to up position stop (Fig. 2). This cams contacts into "open" position. DIP device can then be dropped into open contacts. In Fig. 3 DIP is pushed downward and contacts begin to close. When tip of device lead is past contact point (Fig. 4), contacts close and wipe on lead for remaining distance.

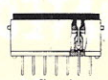
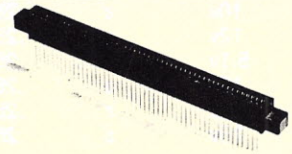


Figure 4



### S-100 BUS CONNECTORS (IMSAI TYPE)

2x50x.125 x .25 Rows.

Gold, Solder tail for Mother boards $4.50,4/$17.00
Tin-Nickel, (NASGLO) Solder tail $3.75,4/$14.00
Gold, wire-wrap.............. $4.50,4/$17.00
Tin-Nickel, (NASGLO) wire-wrap $3.75,4/$14.00

### I.C. SOCKETS

Lo Profile Tin Solder Tail Dip Sockets

| | | | |
|---|---|---|---|
| 8 pin | 10/$1.50 | 100/$14.00 | 1000/$120.00 |
| 14 pin | 10/$1.70 | 100/$16.00 | 1000/$140.00 |
| 16 pin | 10/$1.90 | 100/$18.00 | 1000/$160.00 |

### VOLTAGE REGULATORS

| | | |
|---|---|---|
| 7805-06-08-12-15-24 TO220 | 95¢ | 5/$4.50 |
| 7905-06-08-12-15-24 TO 220 | 95¢ | 5/$4.50 |
| 78L05A-12-15 4% 100 mA TO-92 Plastic | | 50¢ |
| 78H05KC 5V 5A TO-3 | | 9.15 |
| 78H12KC 12V 5A TO-3 | | 9.15 |
| 78H15KC 15V 5A TO-3 | | 9.15 |
| Lm317K 1.5A *Adjustable* TO-3 | | 4.99 |
| Lm317T 1.5A *Adjustable* TO-220 | | 3.99 |
| Lm317MP .5A *Adjustable* TO-202 | | 13.95 |
| TL430C Adjustable Zener-*Think About It* | | 1.50 |
| TL497C Switching Reg. & Inductor | | 9.50 |
| RCA CA 3085 100 mA *Adjustable* | | .60 |

Adjustable 3A Regulator. LM350 is an adjustable 3-terminal regulator capable of supplying in EXCESS of 3 Amps over a 1.2V to 33V range. Standard TO-3 package for easy mounting. Full protection from thermal, current and safe area overload. Completely floating and can be used to regulate at any voltage level as long as input-output difference is less than 35Volts. 0.1% regulation!
LM350K.................................. $10.95
Specs...................................... .80

### DIODES AND BRIDGES

| | | |
|---|---|---|
| IN4003 200 V 1 amp | | 12/$1.00 |
| IN4004 400 V 1 amp | | 10/$1.00 |
| IN4148 Hi Speed Signal | 15/$1.00 | 100/$5.00 |
| D-600 115 V, 100 mA Hi Speed Signal | | 20/$1.00 |
| D2131 200 V, 25A Stud | | 85¢ |
| D2135 400 V, 25A Stud | | 1.00 |
| D2138 600 V, 25A Stud | | 1.55 |
| D3289R 200 V, 160A Stud Anode | | 5.85 |
| D3909-4 50 V, 45A Fast Recovery | | 2.00 |
| IN4732A-47A 1W 5% Zeners | | 4/$1.00 |
| 13 Assorted *Brand New* Zener Diodes | | 1.00 |
| 50V 3 amp Epoxy Bridge | | 79¢ |
| 200V 30 amp Bridge | | 2.00 |
| 600V 4 amp Epoxy Bridge | | 1.49 |
| 600V 3 amp Stud Bridge | | .89 |
| SI-2 200V, 1.5A Gold Leads | | 15/$1.00 |
| D1A-0030 30V DIAC | | 10/$1.00 |

### MISCELLANEOUS

| | |
|---|---|
| RG-174 Miniature 50 Ω coax | 50'/4.25 |
| WSU-30 Wire Wrap/unwrap tool | 5.95 |
| WSU-30M Modified Wrap/unwrap tool | 6.95 |
| BW-630 Battery Operated Wrap Tool | 34.95 |
| —Free Wire with any Wrap Tool — | |
| Miniature Square .05/100V Monolithic Cap | 10/2.00 |
| 2N4036 90V, IA PNP Silicon TO-5 | .50 |
| 2N6101 80V, 10A NPN HI GAIN TO-220 | .50 |
| SE7005 250V, ½A NPN Silicon TO-5 W/Flange | .60 |
| 6.3 VCT, 1.2A Transformer F41X | 2.49 |
| 12V, 1A Transformer with 6' Power Cord | 2.88 |

Fairchild Linear Data Book. Huge volume has 17 chapters of data, applications and definitions. Probably the biggest volume of its type today......... $4.50ppd

Fairchild Bipolar Memory Data Book. ROMS, PROMS and RAMS are covered in this work with full engineering data. Price includes shipping!........... $3.75

Raytheon Linear Data Book. Covers many of the familiar as well as those for which Raytheon is the inovator. Contains valuable applications data as well. as complete specifications................ $2.95ppd

Raytheon Micro Computer Components Book. Covers Raytheons' offerings in the field from the micro-puter to memories and adapters............... $2.95ppd

tri-tek, inc.
7808 North 27th Avenue
Phoenix, Arizona 85021
(602) 995-9352

T1

- Please give street address for UPS shipping when possible.
- C.O.D. NO parcel post C.O.D.
  UPS C.O.D. Add 85¢ to order.
- Any correspondence not connected with your order, please use separate sheet and include SASE for reply.
- Orders less than $10 ($15 foreign) please add $1 handling.
- Prices are subject to change without notice.
- Any refunds will be by check, not credit vouchers.
- Terms, Check, money order, credit card. Net 30 days to rated firms, schools and government agencies.
- If we should be temporarily out of stock on an item, it will be placed on back order. If we cannot ship in 30 days, you will be notified of the expected shipping date and furnished with a postage paid card with which to cancel your order if desired.
- We pay surface shipping only in USA, Canada and Mexico.
- For premium shipping (first class, special handling, etc.) add extra. Excell will be refunded.
- Foreign orders (except Canada and Mexico) estimate and add shipping. Excess will be refunded.

**Charge card telephone orders ($20 min.) will be accepted 9-5:30 P.M. except weekends. Telephone 995-9352. No collect calls please.**

## DIODES/ZENERS

| | | | |
|---|---|---|---|
| 1N914 | 100v | 10mA | .05 |
| 1N4005 | 600v | 1A | .08 |
| 1N4007 | 1000v | 1A | .15 |
| 1N4148 | 75v | 10mA | .05 |
| 1N753A | 6.2v | z | .25 |
| 1N758A | 10v | z | .25 |
| 1N759A | 12v | z | .25 |
| 1N4733 | 5.1v | z | .25 |
| 1N5243 | 13v | z | .25 |
| 1N5244B | 14v | z | .25 |
| 1N5245B | 15v | z | .25 |

## SOCKETS/BRIDGES

| | | | | |
|---|---|---|---|---|
| 8-pin | pcb .25 | ww | .45 |
| 14-pin | pcb .25 | ww | .40 |
| 16-pin | pcb .25 | ww | .40 |
| 18-pin | pcb .25 | ww | .75 |
| 22-pin | pcb .45 | ww | 1.25 |
| 24-pin | pcb .35 | ww | 1.10 |
| 28-pin | pcb .35 | ww | 1.45 |
| 40-pin | pcb .50 | ww | 1.25 |
| Molex pins .01 | | To-3 Sockets | .45 |
| 2 Amp Bridge | | 100-prv | 1.20 |
| 25 Amp Bridge | | 200-prv | 1.95 |

## TRANSISTORS, LEDS, etc.

| | | |
|---|---|---|
| 2N2222 | NPN (Plastic .10) | .15 |
| 2N2907 | PNP | .15 |
| 2N3906 | PNP | .10 |
| 2N3054 | NPN | .35 |
| 2N3055 | NPN 15A 60v | .50 |
| T1P125 | PNP Darlington | .35 |
| LED Green, Red, Clear | | .15 |
| D.L.747 | 7 seg 5/8" high com-anode | 1.95 |
| XAN72 | 7 seg com-anode | 1.50 |
| FND 359 | Red 7 seg com-cathode | 1.25 |

## C MOS

| | |
|---|---|
| 4000 | .15 |
| 4001 | .20 |
| 4002 | .20 |
| 4004 | 3.95 |
| 4006 | 1.20 |
| 4007 | .35 |
| 4008 | .95 |
| 4009 | .30 |
| 4010 | .45 |
| 4011 | .20 |
| 4012 | .20 |
| 4013 | .40 |
| 4014 | 1.10 |
| 4015 | .95 |
| 4016 | .35 |
| 4017 | 1.10 |
| 4018 | 1.10 |
| 4019 | .60 |
| 4020 | .85 |
| 4021 | 1.35 |
| 4022 | .95 |
| 4023 | .25 |
| 4024 | .75 |
| 4025 | .35 |
| 4026 | 1.95 |
| 4027 | .50 |
| 4028 | .95 |
| 4030 | .35 |
| 4033 | 1.50 |
| 4034 | 2.45 |
| 4035 | 1.25 |
| 4040 | 1.35 |
| 4041 | .69 |
| 4042 | .95 |
| 4043 | .95 |
| 4044 | .95 |
| 4046 | 1.75 |
| 4049 | .70 |
| 4050 | .50 |
| 4066 | .95 |
| 4069 | .40 |
| 4071 | .35 |
| 4081 | .70 |
| 4082 | .45 |

## – T T L –

| | |
|---|---|
| 7400 | .15 |
| 7401 | .15 |
| 7402 | .20 |
| 7403 | .20 |
| 7404 | .15 |
| 7405 | .25 |
| 7406 | .35 |
| 7407 | .55 |
| 7408 | .25 |
| 7409 | .15 |
| 7410 | .10 |
| 7411 | .25 |
| 7412 | .30 |
| 7413 | .45 |
| 7414 | 1.10 |
| 7416 | .25 |
| 7417 | .40 |
| 7420 | .15 |
| 7426 | .30 |
| 7427 | .45 |
| 7430 | .15 |
| 7432 | .30 |
| 7437 | .35 |
| 7438 | .35 |
| 7440 | .25 |
| 7441 | 1.15 |
| 7442 | .45 |
| 7443 | .85 |
| 7444 | .45 |
| 7445 | .65 |
| 7446 | .95 |
| 7447 | .95 |
| 7448 | .70 |
| 7450 | .25 |
| 7451 | .25 |
| 7453 | .20 |
| 7454 | .25 |
| 7460 | .40 |
| 7470 | .45 |
| 7472 | .40 |

| | |
|---|---|
| 7473 | .25 |
| 7474 | .35 |
| 7475 | .35 |
| 7476 | .30 |
| 7480 | .55 |
| 7481 | .75 |
| 7483 | .95 |
| 7485 | .95 |
| 7486 | .30 |
| 7489 | 1.35 |
| 7490 | .55 |
| 7491 | .95 |
| 7492 | .95 |
| 7493 | .40 |
| 7494 | 1.25 |
| 7495 | .60 |
| 7496 | .80 |
| 74100 | 1.85 |
| 74107 | .35 |
| 74121 | .35 |
| 74122 | .55 |
| 74123 | .55 |
| 74125 | .45 |
| 74126 | .35 |
| 74132 | 1.35 |
| 74141 | 1.00 |
| 74150 | .85 |
| 74151 | .75 |
| 74153 | .95 |
| 74154 | 1.05 |
| 74156 | .95 |
| 74157 | .65 |
| 74161 | .85 |
| 74163 | .95 |
| 74164 | .60 |
| 74165 | 1.50 |
| 74166 | 1.35 |
| 74175 | .80 |

| | |
|---|---|
| 74176 | 1.25 |
| 74180 | .85 |
| 74181 | 2.25 |
| 74182 | .95 |
| 74190 | 1.75 |
| 74191 | 1.35 |
| 74192 | 1.65 |
| 74193 | .85 |
| 74194 | 1.25 |
| 74195 | .95 |
| 74196 | 1.25 |
| 74197 | 1.25 |
| 74198 | 2.35 |
| 74221 | 1.00 |
| 74367 | .85 |
| 75108A | .35 |
| 75110 | .35 |
| 75491 | .50 |
| 75492 | .50 |
| 74H00 | .25 |
| 74H01 | .25 |
| 74H04 | .25 |
| 74H05 | .25 |
| 74H08 | .35 |
| 74H10 | .35 |
| 74H11 | .25 |
| 74H15 | .30 |
| 74H20 | .30 |
| 74H21 | .25 |
| 74H22 | .40 |
| 74H30 | .25 |
| 74H40 | .25 |
| 74H50 | .25 |
| 74H51 | .25 |
| 74H52 | .15 |
| 74H53J | .25 |
| 74H55 | .25 |

| | |
|---|---|
| 74H72 | .55 |
| 74H101 | .75 |
| 74H103 | .75 |
| 74H106 | .95 |
| 74L00 | .35 |
| 74L02 | .35 |
| 74L03 | .30 |
| 74L04 | .35 |
| 74L10 | .35 |
| 74L20 | .35 |
| 74L30 | .45 |
| 74L47 | 1.95 |
| 74L51 | .45 |
| 74L55 | .65 |
| 74L72 | .45 |
| 74L73 | .40 |
| 74L74 | .45 |
| 74L75 | .55 |
| 74L93 | .55 |
| 74L123 | .55 |
| 74S00 | .55 |
| 74S02 | .55 |
| 74S03 | .30 |
| 74S04 | .35 |
| 74S05 | .35 |
| 74S08 | .35 |
| 74S10 | .35 |
| 74S11 | .35 |
| 74S20 | .35 |
| 74S40 | .25 |
| 74S50 | .25 |
| 74S51 | .45 |
| 74S64 | .25 |
| 74S74 | .40 |
| 74S112 | .90 |
| 74S114 | 1.30 |

| | |
|---|---|
| 74S133 | .45 |
| 74S140 | .75 |
| 74S151 | .35 |
| 74S153 | .35 |
| 74S157 | .80 |
| 74S158 | .35 |
| 74S194 | 1.05 |
| 74S257 (8123) | .25 |
| 74LS00 | .35 |
| 74LS01 | .35 |
| 74LS02 | .35 |
| 74LS04 | .35 |
| 74LS05 | .45 |
| 74LS08 | .35 |
| 74LS09 | .35 |
| 74LS10 | .35 |
| 74LS11 | .35 |
| 74LS20 | .35 |
| 74LS21 | .25 |
| 74LS22 | .25 |
| 74LS32 | .40 |
| 74LS37 | .35 |
| 74LS40 | .45 |
| 74LS42 | 1.10 |
| 74LS51 | .50 |
| 74LS74 | .65 |
| 74LS86 | .65 |
| 74LS90 | .95 |
| 74LS93 | .95 |
| 74LS107 | .85 |
| 74LS123 | 1.00 |
| 74LS151 | .95 |
| 74LS153 | 1.20 |
| 74LS157 | .85 |
| 74LS164 | 1.90 |
| 74LS367 | .85 |
| 74LS368 | .85 |

## 9000 SERIES

| | |
|---|---|
| 9301 | .85 |
| 9309 | .35 |
| 9322 | .85 |
| 95H03 | .55 |
| 9601 | .75 |
| 9602 | .50 |

## MEMORY CLOCKS

| | |
|---|---|
| 74S188 (8223) | 3.00 |
| 1702A | 6.95 |
| MM5314 | 3.00 |
| MM5316 | 3.50 |
| 2102-1 | 1.75 |
| 2102L-1 | 1.95 |
| TR 1602B/ TMS 6011 | 6.95 |
| 8080AD | 15.00 |
| 8T13 | 1.50 |
| 8T23 | 1.50 |
| 8T24 | 2.00 |
| 2107B-4 | 4.95 |

## LINEARS, REGULATORS, etc.

| | |
|---|---|
| 8266 | .35 |
| MCT2 | .95 |
| 8038 | 3.95 |
| LM201 | .75 |
| LM301 | .25 |
| LM308 (Mini) | .75 |
| LM309H | .65 |
| LM309K (340K-5) | .85 |
| LM310 | 1.15 |
| LM311D (Mini) | .75 |
| LM318 (Mini) | .65 |

| | |
|---|---|
| LM320K5 (7905) | 1.65 |
| LM320K12 | 1.65 |
| LM320T5 | 1.65 |
| LM320T12 | 1.65 |
| LM320T15 | 1.65 |
| LM339 | .95 |
| 7805 (340T5) | .95 |
| LM340T12 | 1.00 |
| LM340T15 | 1.00 |
| LM340T18 | 1.00 |

| | |
|---|---|
| LM340T24 | .95 |
| LM340K12 | 2.15 |
| LM340K15 | 1.25 |
| LM340K18 | 1.25 |
| LM340K24 | .95 |
| LM373 | 2.95 |
| LM380 | .95 |
| LM709 (8,14 PIN) | .25 |
| LM711 | .45 |

| | |
|---|---|
| LM723 | .50 |
| LM725 | 1.75 |
| LM739 | 1.50 |
| LM741 (8-14) | .25 |
| LM747 | 1.10 |
| LM1307 | 1.25 |
| LM1458 | .95 |
| LM3900 | .50 |
| LM75451 | .65 |
| NE555 | .50 |
| NE556 | .95 |
| NE565 | .95 |
| NE566 | 1.75 |
| NE567 | 1.35 |

# INTEGRATED CIRCUITS UNLIMITED

**7889 Clairemont Mesa Boulevard, San Diego, California 92111**
**(714) 278-4394** *(Calif. Res.)*

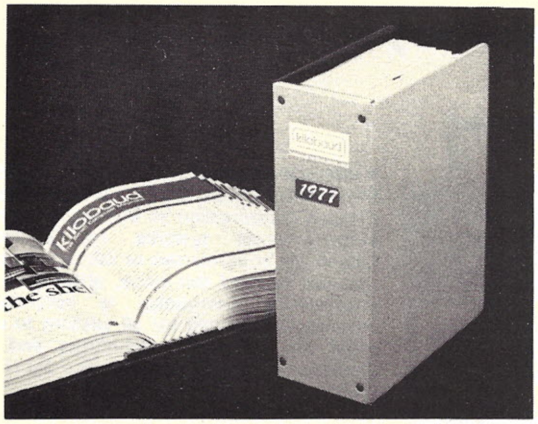
All orders shipped prepaid — No minimum
Open accounts invited — COD orders accepted

Discounts available at OEM Quantities — California Residents add 6% Sales Tax
All IC's Prime/Guaranteed. All orders shipped same day received.
**24 Hour Toll Free Phone 1-800-854-2211** — **MasterCharge / BankAmericard / AE**



## SPECIAL DISCOUNTS

| Total Order | Deduct |
|---|---|
| $35 - $99 | 5% |
| $100 - $300 | 10% |
| $301 - $1000 | 15% |
| $1000 - Up | 20% |



# 1977 kilobaud Binders

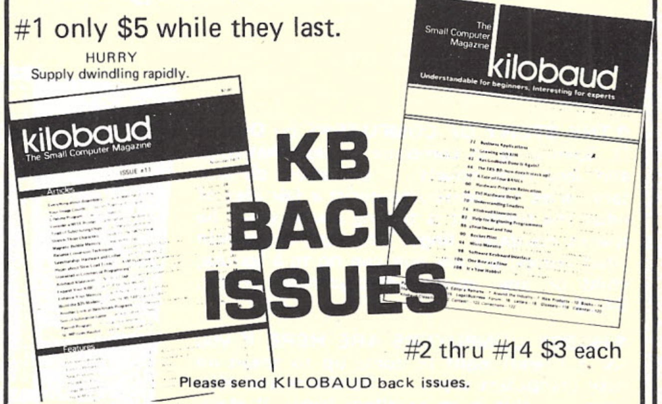
Keep your Kilobauds from being lost or damaged in these handsomely appointed binders with rich blue covers and gold lettering. Each binder holds 12 issues, making an EXCELLENT REFERENCE HANDBOOK. Several binders form a quality library you can be proud of.

Order yours today . . . only $6.00 each . . . $5.00 each additional.

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Binders Peterborough NH 03458
Be sure to include check or detailed credit card information
*Add $1.00 shipping & handling charge for each order.*

**Did you miss out on the first issue of Kilobaud?**
**Don't lose your chance to get them NOW!**

#1 only $5 while they last.
HURRY
Supply dwindling rapidly.




# KB BACK ISSUES

#2 thru #14 $3 each

Please send KILOBAUD back issues.

| | | |
|---|---|---|
| ___ issues JAN 77 $5 | ___ issues JUNE 77 $3 | ___ issues NOV 77 $3 |
| ___ issues FEB 77 $3 | ___ issues JUL 77 $3 | ___ issues DEC 77 $3 |
| ___ issues MAR 77 $3 | ___ issues AUG 77 $3 | ___ issues JAN 78 $3 |
| ___ issues APR 77 $3 | ___ issues SEPT 77 $3 | ___ issues FEB 78 $3 |
| ___ issues MAY 77 $3 | ___ issues OCT 77 $3 | ___ TOTAL |

☐ BankAmericard ☐ Master Charge ☐ American Express

Card # ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration date__________

Signature__________

Name__________

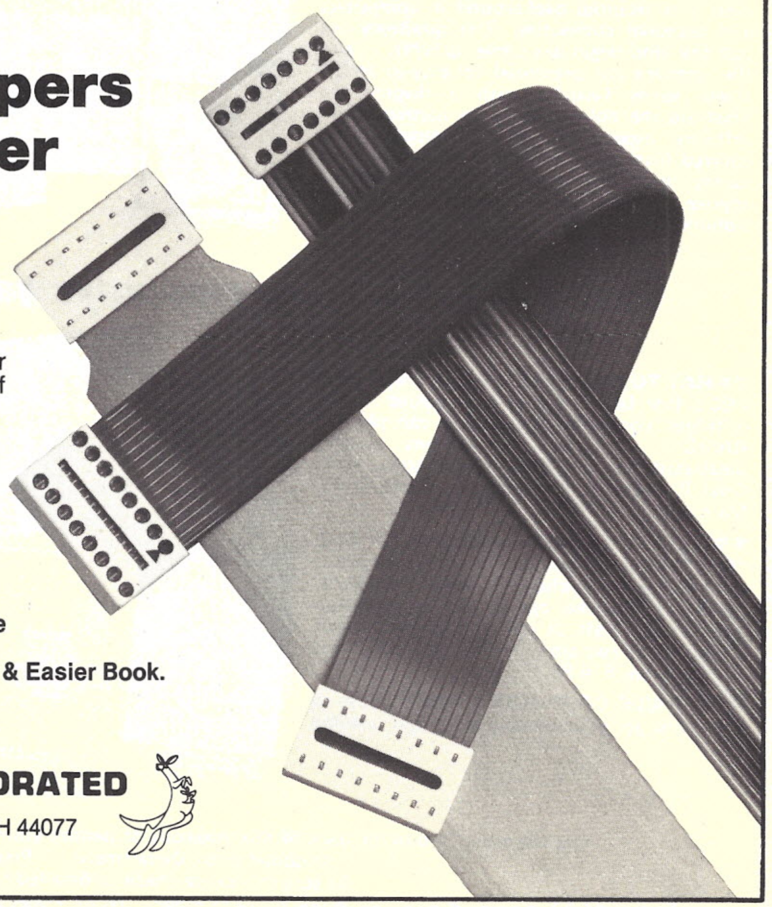
Address__________

City__________ State______ Zip______

*Add $1.00 shipping & handling charge for each order

Have credit card # handy and call
TOLL FREE (800) 258-5473 KB/3/78
KILOBAUD • PETERBOROUGH NH 03458

# At last...DIP Jumpers for Faster & Easier connections.



AP DIP Jumpers are the low cost high quality solution to bussing between PC boards, mother boards, backplanes and more. Available in 14, 16, 24 and 40 pin single ended or double ended assemblies, in standard lengths of 6, 12, 24 and 36 inches. Each assembly has molded-on strain relief and line-by-line probeability. Contact material is non-corrosive nickel silver.Dielectric is 94 V-O rated. Cable options include stranded electric pink, rainbow or with ground plane.

**Order from your AP distributor today. Our distributor list is growing daily. For the name of the distributor nearest you call Toll Free 800-321-9668.**

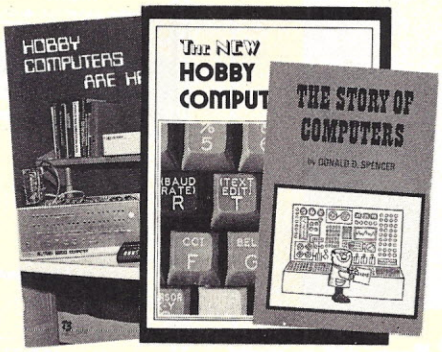
**Send for our complete AP catalog The Faster & Easier Book.**

***Faster & Easier is what we're all about.***

**AP PRODUCTS INCORPORATED**
Box 110 • 72 Corwin Drive, Painesville OH 44077
(216) 354-2101 TWX: 810-425-2250

KB book nook KB book nook

## beginner's•introductory

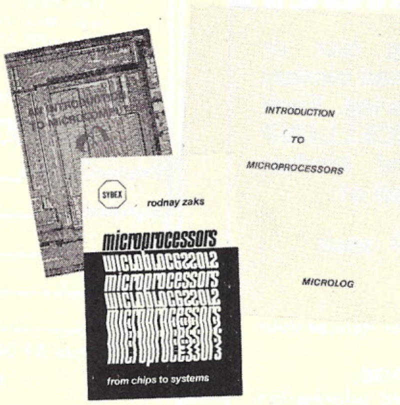



•**THE STORY OF COMPUTERS** by Donald D. Spencer is to computer books what *Dick and Jane* is to novels . . . extremely elementary, gives the non-computerist a fair idea of what the hobbyist is talking about when he speaks computer lingo. Attempts to explain what computers are and can do to a spouse, child or any un-electronics-minded friend. $4.95.*

•**HOBBY COMPUTERS ARE HERE** If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, the basics of programming, along with a couple of TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95.*

•**THE NEW HOBBY COMPUTERS!** This book takes it from where "Hobby Computers Are Here" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards, a Baudot monitor/editor system, an audible logic probe for finding those tough problems, a ham's computer, a computer QSO machine . . . and much, much more! $4.95.*

•**HOME COMPUTERS: $2^{10}$ Questions & Answers** by Rich Didday. Two books aimed exclusively at the novice computer hobbyist/home computer user. Written in a rather unusual style which has a beginner asking questions which are answered by a person with a substantial background in computers and personal computing. The questions are just the kind beginners come up with . . . and the answers are presented in easy-to-understand terms (usually with a diagram to illustrate the point). Both the hardware and software aspects of home computing are covered from A to Z. An index in both books makes them ideal as reference material *for anyone.* Volume I: Hardware — $7.95*; Volume 2: Software — $6.95*.





•**MICROPROCESSORS FROM CHIPS TO SYSTEMS** by Rodnay Zaks is a complete and detailed introduction to microprocessors and microcomputer systems. No preliminary knowledge of computers or microprocessors is required to read this book, although a basic engineering knowledge is naturally an advantage. Intended for all wishing to understand the concepts, techniques and components of microprocessors in a short time. $9.95.*

•**MICROCOMPUTER PRIMER** by Mitchell Waite and Michael Pardee. Describes basic computer theory, explains numbering systems, and introduces the reader to the world of programming. Describes the world of microcomputing in "real world" terminology. No better way of getting involved with the exciting new hobby of microcomputing. $7.95.*

•**INTRODUCTION TO MICROPROCESSORS** by Charles Rockwell of MICROLOG is an ideal reference for the individual desiring to understand the hardware aspects of microprocessor systems. Describes the hardware details of computer devices in terms the beginner can understand, instead of treating the micro chip as a "black box." General information about hardware systems is provided. Specific systems are not described and programming is only briefly discussed. $17.50 US and Canada, $20 elsewhere.*

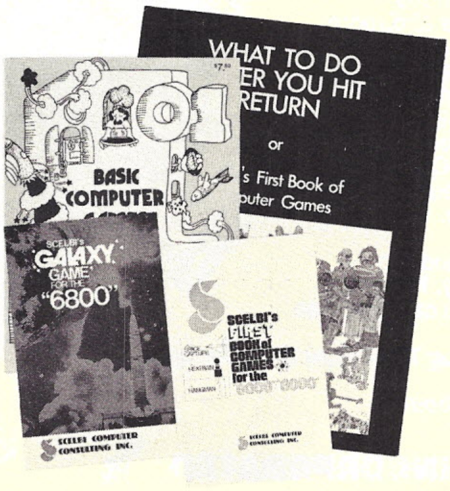
•**AN INTRODUCTION TO MICROCOMPUTERS, VOLS. 1 AND 2** by Adam Osborne Associates, are references dealing with microcomputer architecture in general and specifically with details about most of the common chips. These books are not software-oriented, but are invaluable for the hobbyist who is into building his own interfaces and processors. Volume 1 is dedicated to general hardware theory related to micros, and Volume 2 discusses the practical details of each micro chip. (Detailed review in Kilobaud #2) Volume 1 — $7.50*; Volume II — $12.50.*

## computer games

•**WHAT TO DO AFTER YOU HIT RETURN** PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, muchly illustrated. Lunar landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $8.00.*

•**SCELBI'S GALAXY GAME FOR THE "6800"** Here's a new twist in computer games by Robert Findley/Raymond Edwards. "Galaxy" pits the operator of a spaceship against alien craft, as well as such variables as speed, time, and ammunition. No two games are the same! $14.95.*

•**101 BASIC COMPUTER GAMES** Okay, so once you get your computer up and running




in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you, from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

•**SCELBI'S FIRST BOOK OF COMPUTER GAMES** Need a game for your 8008 or 8080 microprocessor? Has three popular games, "Space Capture," "Hexpawn," and "Hangman." Complete flowcharts, logic description, program listing, and instructions are provided. A must for the game freak! $14.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department Peterborough NH 03458
Be sure to include check or detailed credit card information

* *Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.

# software·programming

●**DISCOVERING BASIC — A Problem Solving Approach** by Robert E. Smith deals with progressively more complex problems which allow the reader to discover the vocabulary of BASIC language as he develops skill and confidence in putting it to work. Clear and concise explanations. Problems used cover a wide range of interests — insurance, geometry, puzzles, economics, etc. $6.85.*

●**BASIC** by Bob Albrecht, etc. Self-teaching guide to the computer language you will need to know for use with your microcomputer. This is one of the easiest ways to learn computer programming. $4.95.*

●**A QUICK LOOK AT BASIC** by Donald D. Spencer. A perfect reference for the beginning programmer. Assumes that the reader has no previous programming experience and is a self-teaching guide for the individual desiring to learn the fundamentals of BASIC. $4.95.*

●**MY COMPUTER LIKES ME . . . WHEN I SPEAK BASIC** An introduction to BASIC . . . simple enough for your kids. If you want to teach BASIC to anyone quickly, this book is the way to go. $2.00.*

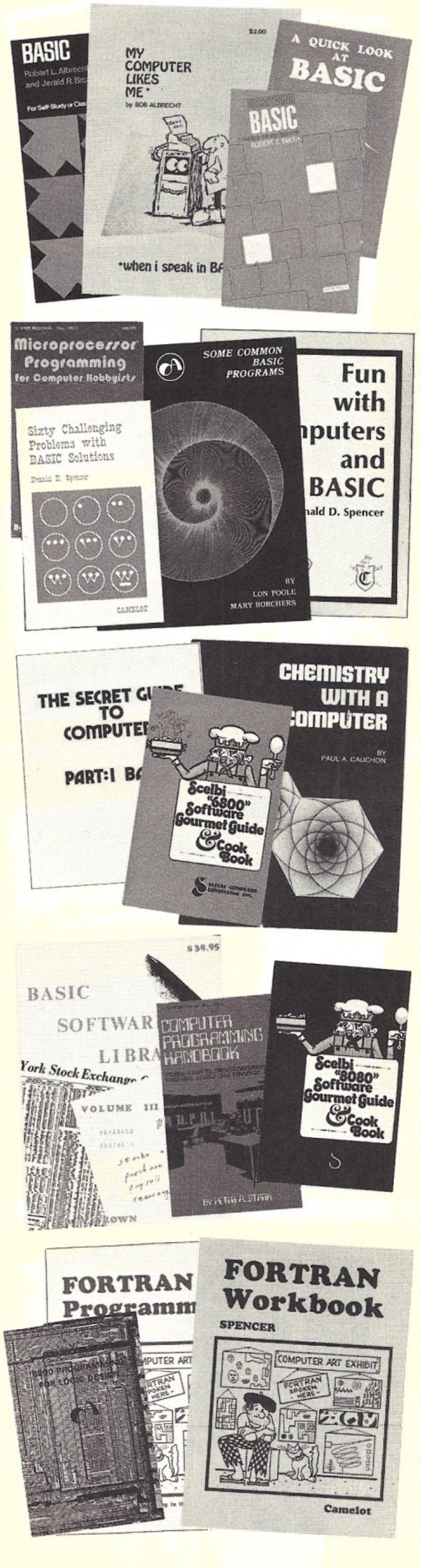
●**FUN WITH COMPUTERS AND BASIC** by Donald D. Spencer, contains an easy-to-understand explanation of the BASIC Programming Language and is intended for persons who have had no previous exposure to computer programming. Over half the book is devoted to problems using games, puzzles, and mathematical recreations. A superior book for self-teaching and learning computer programming. $6.95.*

●**SIXTY CHALLENGING PROBLEMS WITH BASIC SOLUTIONS** by Donald Spencer, provides the serious student of BASIC programming with interesting problems and solutions. No knowledge of math above algebra required. Incudes a number of game programs, as well as programs for financial interest, conversions and numeric manipulations. $6.95.*

●**THE SECRET GUIDE TO COMPUTERS** Parts 1, 2, and 3 by Russ Walter. **Part One** describes computers in general, and after reading for ten minutes you will be writing simple BASIC programs! **Part Two** discusses computer applications. It's one thing to master the syntax of a language such as BASIC and another to solve problems using the new tool. **Part Three** describes programming languages. Ever heard of APL and QLISP? BASIC is not the only language used to program computers. 7th Edition. Part I - $2.75*; Part II — $2.50*; Part III — $3.50.*

●**SOME COMMON BASIC PROGRAMS** published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the stand-alone mode. $7.50 paperback.*

●**Scientific Research Instruments' BASIC SOFTWARE LIBRARY** is a complete do-it-yourself kit. Written in everybody's BASIC immediately executable in ANY computer with at least 4K, no other peripherals needed. Vol. I contains business and recreational programs and is 300 pages. Vol. II is 260 pages and contains math, engineering, statistics and plotting programs. Vol. III contains money managing, advanced business programs such as billing, A/R, inventory, payroll, etc. Vol. IV contains general purpose programs like loans, rates, retirement, plus games: Poker, Enterprise (take charge while Capt. Kirk is away), Football and more! Vol. V is filled with experimenter's programs including games, pictures and misc. problems like "logic." Vols. I & II $24.95,* Vol. III $39.95,* Vol. IV & V $9.95 each.*




●**MICROPROCESSOR PROGRAMMING for Computer Hobbyists** by Neill Graham is for the hobbyist interested in intermediate and advanced techniques of programming and data structuring. Written to take up where the computer manufacturers' instruction manuals and the introductory programming language texts leave off. $8.95.*

● **ADVANCED BASIC — Applications and Problems** by James Coan is for those who want to extend their expertise with BASIC. Offers advanced techniques and applications. $6.95*

● **8080 PROGRAMMING FOR LOGIC DESIGN** Ideal reference for an in-depth understanding of the 8080 processor. Application-oriented and the 8080 is discussed in light of replacing conventional, hard-wired logic. Practical design considerations are provided for the implementation of an 8080-based control system. $7.50.*

●**8080 SOFTWARE GOURMET GUIDE AND COOKBOOK** If you have been spending too much time developing simple routines for your 8080, try this new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 8080 user. $9.95.*

● **6800 PROGRAMMING FOR LOGIC DESIGN** Oriented toward the industrial user, this book describes the process by which conventional logic can be replaced by a 6800 microprocessor. Provides practical information that allows an experimenter to design a complete micro control system for the "ground up." $7.50.*

●**6800 SOFTWARE GOURMET GUIDE & COOKBOOK** If you have been spending too much time developing routines for your 6800 microprocessor, try the new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 6800 user. $9.95.*

●**FORTRAN PROGRAMMING** by Donald Spencer. FORTRAN was designed for complex numeric calculations; and possesses extended I/O capability. It is easily learned, as it is an English-like computer language. $7.50.*

● **FORTRAN WORKBOOK** by Donald Spencer. Provides practical examples and problems to solve. Flowcharting is also discussed. Almost all micros support BASIC — it won't be long before FORTRAN is commonplace. $3.95.*

●**CHEMISTRY WITH A COMPUTER** by Paul A. Cauchon, contains a collection of tutorial, simulation and problem-generation computer programs. Usable with almost any chemistry course in the high school or college level. $9.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department Peterborough NH 03458
Be sure to include check or detailed credit card information

* *Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.

KB book nook KB book nook

KB book nook

## hardware


● **MICROPROCESSOR INTERFACING TECHNIQUES** by Austin Lesea & Rodnay Zaks will teach you how to interconnect a complete system and interface it to all the usual peripherals. It covers hardware and software skills and techniques, including the use and design of model buses such as the IEEE 488 or S100. $9.95.*

● **TTL COOKBOOK** by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter, and a digital tachometer. $8.95.*

● **CMOS COOKBOOK** by Don Lancaster. Details the application of CMOS, the low power logic family suitable for most applications presently dominated by TTL. Required reading for every serious digital experimenter! $9.95.*

● **TVT COOKBOOK** by Don Lancaster, describes the use of a standard television receiver as a microprocessor CRT terminal. Explains and describes character generation, cursor control and interface information in typical, easy-to-understand Lancaster style. $9.95.*

● **BUILD-IT BOOK OF DIGITAL ELECTRONIC TIMEPIECES** by Robert Haviland is a data-packed guide to building every timekeeping device you can imagine: rugged shipboard clocks, second-splitting digital IC chronometers, decorator digital clocks, a precision timer, a frequency-period meter, a tide and moon clock, an automatic alarm setter, etc. Including full-size printed circuit board layouts. $6.95.*

● **THE "COMPULATOR" BOOK — Building Super Calculators & Minicomputer Hardware with Calculator Chips** by R. P. Haviland, provides ideas, design info and printed circuit boards for calculator chip projects like "training" an inexpensive "garden variety" calculator chip to dial a telephone, measure time, tie in with a Teletype to create a virtually infinite memory system, and countless other functions. $7.95.*

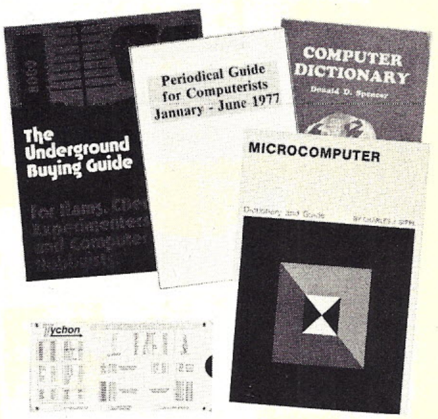
● **VOL I COMPONENT TESTERS** Build your own test equipment and save a bundle (and have a lot of fun). Volume I of the 73 Test Equipment Library shows you how to build and use transistor testers (8 of 'em), three diodes testers, 3 IC testers, 9 voltmeters and VTVMs, 8 ohmmeter, 3 inductance meters, and a raft of other gadgets for checking temperature, crystals, Q, etc. $4.95.*

● **VOL II AUDIO FREQUENCY TESTERS** If you're into audio such as digital cassette recording, RTTY, Baudot vs ASCII, SSTV, SSB, Touchtone or even hi-fi you'll want to have this book full of home built test equipment projects. Volume II $4.95.*

● **VOL III RADIO FREQUENCY TESTERS** This is of more interest to hams and CBers: test equipment you can build for checking out transmitters and receivers, signal generators, noise generators, crystal calibrators, GDOs, dummy loads. $4.95.*

● **VOL. IV IC TEST EQUIPMENT** Become a trouble-shooting wizard. All you need to know about pulse, audio and sync generators, frequency counters, digital component testers, logic probes and more! Plus a cumulative index for all four volumes of the 73 Test Equipment Library. $4.95.*

## general




● **COMPUTER DICTIONARY** by Donald D. Spencer. Defines words and acronyms used by computerists in a clear, easy to understand style. This reference is a must for the individual getting started in the world of microcomputers. $5.95.*

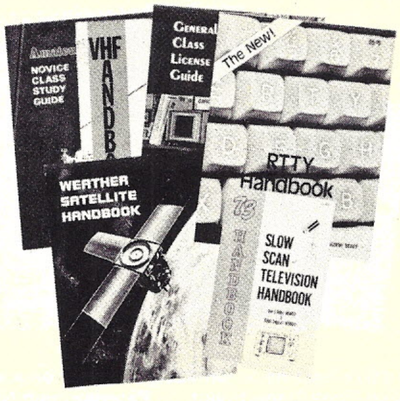
● **BRAND NEW DICTIONARY** This microcomputer dictionary fills the need to become quickly acquainted with the terminology and nomenclature of the revolution in computers. There is also a comprehensive electronics/computer abbreviations and acronyms section. $15.95.*

● **THE UNDERGROUND BUYING GUIDE** Here is a handy guide for the electronics enthusiast. Over 600 sources of equipment and literature are provided. Cross-referenced for ease of use. Electronic publishing houses are also listed. $5.95 each.*

● **1976 PERIODICAL GUIDE FOR COMPUTERISTS** is a 20-page book which indexes over 1,000 personal computing articles for the entire year of 1976 from Byte, Creative Computing, Digital Design, Dr. Dobbs Journal, EDN, Electronic Design, Electronics, Interface Age, Microtrek, Peoples Computer Company, Popular Electronics, QST, Radio Electronics, SCCS Interface and 73 Amateur Radio. Price $3.00.* New January — June 1977 Edition (includes Kilobaud) — $3.00.*

● **TYCHON'S 8080 OCTAL CODE CARD** Slide rule-like aid for programming and debugging 8080 software contains all the mnemonics and corresponding octal codes. Also available, Tychon's *8080 Hex Code Card*, same as above only has hex codes instead of octal. $3.00 each.*

## amateur radio books




● **NOVICE STUDY GUIDE** The most complete Novice study guide available. It is brand new. This is not only invaluable for anyone wanting to get started in amateur radio, but also it is about the only really simple book on the fundamentals of electricity and electronics. $4.95.*

● **GENERAL CLASS STUDY GUIDE** Takes over on theory where the Novice book leaves off. You'll need to know the electronic theory in this to work with computers and you'll not find an easier place to get the information. $5.95.*

● **SSTV HANDBOOK** This excellent book tells all about it, from its history and basics to the present state-of-the-art techniques. Hardbound $7,* Softbound $5.*

● **VHF ANTENNA HANDBOOK** This new handbook details the theory, design and construction of hundreds of different VHF and UHF antennas. Packed with fabulous antenna projects you can build. $4.95.*

● **WEATHER SATELLITE HANDBOOK** Simple equipment and methods for getting good pictures from the weather satellite. Dr. Taggart WB8DQT $4.95.*

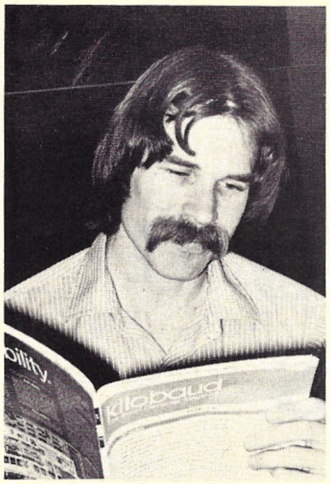
● **THE NEW RTTY HANDBOOK** is a brand new 1977 edition and the only up-to-date RTTY book available. The state-of-the-art has been changing radically and has made all previous RTTY books obsolete. It has the latest circuits, great for the newcomer and the expert alike. $5.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department Peterborough NH 03458 Be sure to include check or detailed credit card information

*Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.

KB book nook

# WHO'S BEHIND THE KILOBAUD?




John Zalabak comes out from behind the Kilobaud.

**WHO'S BEHIND THE KILOBAUD?**

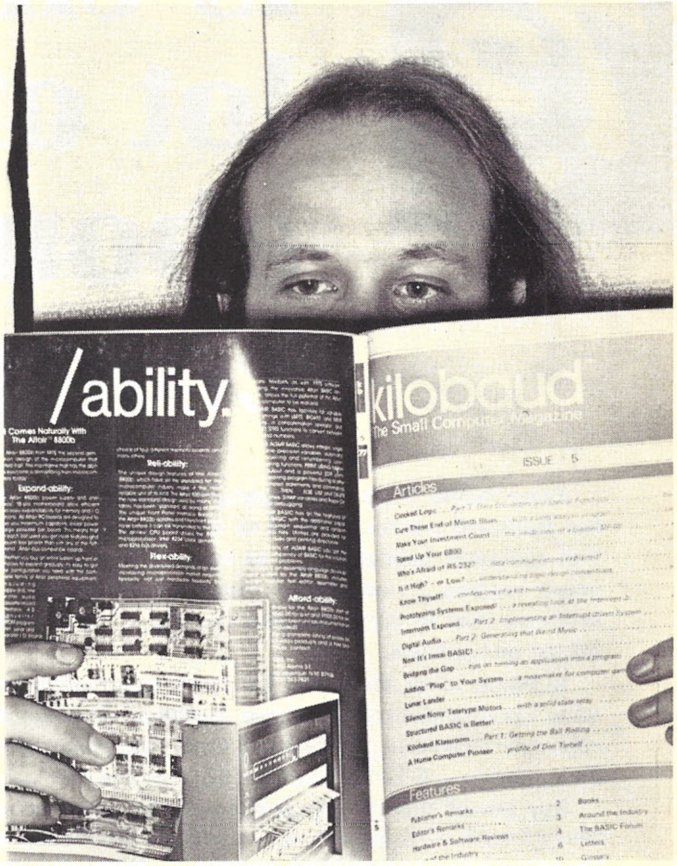
In between selling subscriptions and seeing the other exhibits at computer shows, Wayne has been snapping some pictures of people you have been seeing and will be seeing at shows. How many of them can you recognize behind the Kilobauds?

On page 143 of the February issue there was a picture of John Zalabak of Parasitic Engineering. Okay . . . so you missed that one. Now that you know who he is, be sure to get together with John at your next show.

Kilobaud is the most read magazine in the microcomputer field, so it hasn't been difficult to find key industry people reading the magazine.

**kilobaud**

**PETERBOROUGH NH 03458**




WHO'S THIS

Above is another partly hidden man that you've seen . . . if you've been to any shows. You want a hint, right? Think software, chum . . . or wait until the unveiling next month in KILOBAUD.

# READ IT YOURSELF!

If you are not yet a subscriber to Kilobaud, we need you and you need us. There is a lot of data in Kilobaud that you will want to have on hand . . . the magazine is like a continuing encyclopedia of microcomputing and programs. You never know when you are desperately going to need something from a past issue . . . and you'll want it immediately. The cost per year is not significant . . . $15 at present for $24 worth of magazines.

You may have noticed that Kilobaud has more articles than any of the other magazines. If you'll keep track of how long it takes you to read Kilobaud as compared to the other computer magazines, you'll see you are getting a great bargain at $15 per year.

## SUBSCRIBE TODAY!

**For Instant Subscriptions Call Our Toll Free Number**

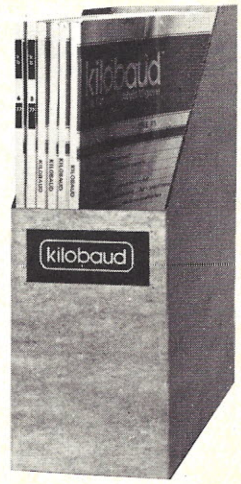
1-(800) 258-5473

**Have your credit card handy!**

**Kilobaud ● Peterborough NH 03458**

## IS HARD COPY STORAGE A PROBLEM?

*Kilobaud*, as thick as it is, is more like a floppy when it comes to standing on the bookshelf. Try the Kilobaud Library Shelf Boxes, . . . sturdy corrugated cardboard boxes which will hold your magazines on the shelf and keep them from flopping around. We have self-sticking labels for the boxes, too, not only for *Kilobaud*, but also for *73 Magazine* . . . and for *Personal Computing*, *Radio Electronics*, *Popular Electronics*, *Interface Age*, and *Byte*. Just ask for whatever stickers you want and we'll throw them in with your box order. Hams may want our labels for *CQ*, *QST* or *Ham Radio*, if they get any of these magazines. The boxes are a white color and are resistant to dirt. They hold a full year of *Kilobaud* . . . or *73*. You can put the boxes on your shelves with the bindings showing or with just the white board showing; there are marks to help you center your labels on either side. Your magazine library is your prime reference, so keep it handy and keep it neat with these strong library shelf boxes . . . $2.00 for the first box and $1.50 for each additional box. Be sure to specify which labels we should send. Have your credit card handy and call our toll-free order number 800-258-5473, or use the order card in the back of the magazine and mail to




KILOBAUD LIBRARY SHELF BOXES Peterborough, NH 03458

## COMPUTER CLUBS!
## Do you need to start a club library?

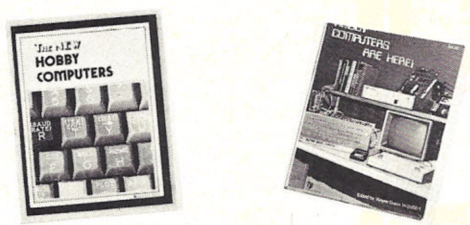
*Start with the BEST!*
Have the secretary of your club send us your current club roster, complete with names, addresses and zip codes and we'll send you one of your choice of the following books, . . . FREE!

**Hobby Computers Are Here**
**The New Hobby Computers**
**IC Test Equipment, the new Vol. IV of the 73 Test Equipment Library**

*(limit one book per club)*

**Kilobaud Clubs**
**Peterborough, NH 03458**

## SEND FOR THESE GREAT BOOKS TODAY!




**• HOBBY COMPUTERS ARE HERE** If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, the basics of programming, along with a couple of TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95

**• THE NEW HOBBY COMPUTERS!** This book takes it from where "Hobby Computers Are Here" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards, a Baudot monitor/editor system, an audible logic probe for finding those tough problems, a ham's computer, a computer QSO machine . . . and much, much more! Everything of interest is there in one volume. Don't miss this tremendous value! Only $4.95

*Have your credit card handy and call toll-free 800-258-5473, or use the card in the back of this magazine and send your check to:*

**KILOBAUD Book Dept. • Peterborough, NH 03458**
*(Be sure to include $1.00 for shipping & handling)*

## MOVING?

Let us know 8 weeks in advance so that you won't miss a single issue of *Kilobaud*.
Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly. Write to:

**• KILOBAUD • PETERBOROUGH NH 03458 •**

☐ Address change only
☐ Extend subscription
☐ Enter new subscription
☐ 1 year $15.00
☐ Payment enclosed (1 extra BONUS issue)
☐ Bill me later

*If you have no label handy, print OLD address here.*

*AFFIX LABEL*

Name __________ Call ______
Address __________
City ______ State ______ Zip ______

*print NEW address here:*

Name __________ Call ______
Address __________
City ______ State ______ Zip ______

# why the last bus is the best bus

If you want to be better, you have to be willing to make changes. So, when we started designing our H8 Computer back in 1975, we put aside a lot of pre-conceptions to design a *better* computer. For instance, instead of slavishly adopting an inferior bus as a "standard," we designed our own. We used fifty fully-buffered lines because that's all we really needed with the H8's built-in systems controller. The result? Clean signals, minimum noise and great expansion capabilities.

Next, we picked a male/female connector. Not only is it less expensive than edge connectors, our tests show it's also more reliable! And because it costs less, every H8 motherboard includes all the connectors you need for expansion. There's nothing extra to buy.

The connectors are mounted on the right side of the cabinet at an angle so the boards tilt back. That gives us a couple of significant advantages. First, it allows the boards to be larger without requiring a taller cabinet. So our circuit board layout is more open, which makes the boards easier to build – even if you've never built a kit before. Second, the angled mounting provides easy accessibility to all the components on the face of the boards without an extender. (Many times an extender alters the transmission characteristics of the bus, which masks the problem you're trying to solve or even causes new ones.)

And we plan to support the H8 bus. In the future you can expect to see many new and exciting accessories for the H8 bus like higher density, lower power memories and a variety of unique I/O interfaces.

We're very happy with our bus, and we think you will be, too.

Who minds being different... as long as you're better.

## Heathkit Computers

# FREE Heathkit Catalog


Read about nearly 400 money-saving, fun-to-build electronic kits.

Use coupon to send for your mail order catalog or bring coupon to a Heathkit Electronic Center for your catalog.

HEATH
Schlumberger

Heath Company, Dept. 351-390
Benton Harbor, MI 49002

H5

Please send me my FREE Catalog. I am not on your mailing list.

Name________________________________

Address________________________________

City____________________ State__________

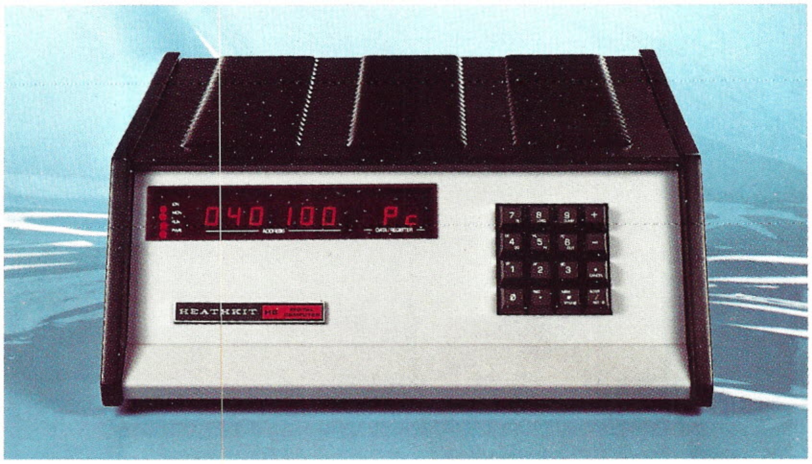
CP-138 Zip__________


**AVAILABLE LOCALLY AT HEATHKIT ELECTRONIC CENTERS**

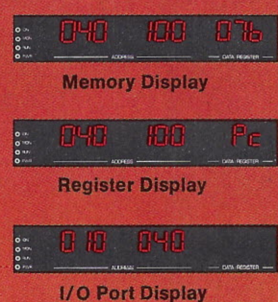
(Units of Schlumberger Products Corporation) Retail prices on some products may be slightly higher.

**ARIZONA: Phoenix, 85017,** 2727 W. Indian School Rd., Phone: 602-279-6247; **CALIFORNIA: Anaheim, 92805,** 330 E. Ball Rd., Phone: 714-776-9420; **El Cerrito, 94530,** 6000 Potrero Ave., Phone: 415-236-8870; **Los Angeles, 90007,** 2309 S. Flower St., Phone: 213-749-0261; **Pomona, 91767,** 1555 Orange Grove Ave. N., Phone: 714-623-3543; **Redwood City, 94063,** 2001 Middlefield Rd., Phone: 415-365-8155; **Sacramento, 95825,** 1860 Fulton Ave., Phone: 916-486-1575; **San Diego (La Mesa, 92041),** 8363 Center Dr., Phone: 714-461-0110; **San Jose (Campbell, 95008),** 2350 S. Bascom Ave., Phone: 408-377-8920; **Woodland Hills, 91364,** 22504 Ventura Blvd., Phone: 213-883-0531; **COLORADO: Denver, 80212,** 5940 W. 38th Ave., Phone: 303-422-3408; **CONNECTICUT: Hartford (Avon, 06001),** 395 W. Main St. (Rte. 44), Phone: 203-678-0323; **FLORIDA: Miami (Hialeah, 33012),** 4705 W. 16th Ave., Phone: 305-823-2280; **Tampa, 33614,** 4019 West Hillsborough Ave., Phone: 813-886-2541; **GEORGIA: Atlanta, 30342,** 5285 Roswell Rd., Phone: 404-252-4341; **ILLINOIS: Chicago, 60645,** 3462-66 W. Devon Ave., Phone: 312-583-3920; **Chicago (Downers Grove, 60515),** 224 Ogden Ave., Phone: 312-852-1304; **INDIANA: Indianapolis, 46220,** 2112 E. 62nd St., Phone: 317-257-4321; **KANSAS: Kansas City (Mission, 66202),** 5960 Lamar Ave., Phone: 913-362-4486; **KENTUCKY: Louisville, 40243,** 12401 Shelbyville Rd., Phone: 502-245-7811; **LOUISIANA: New Orleans (Kenner, 70062),** 1900 Veterans Memorial Hwy., Phone: 504-722-6321; **MARYLAND: Baltimore, 21234,** 1713 E. Joppa Rd., Phone: 301-661-4446; **Rockville, 20852,** 5542 Nicholson Lane, Phone: 301-881-5420; **MASSACHUSETTS: Boston (Peabody, 01960),** 242 Andover St., Phone: 617-531-9330; **Boston (Wellesley, 02181),** 165 Worcester Ave. (Rt. 9 just west of Rt. 128), Phone: 617-237-1510; **MICHIGAN: Detroit, 48219,** 18645 W. Eight Mile Rd., Phone: 313-535-6480; **E. Detroit, 48021,** 18149 E. Eight Mile Rd., Phone: 313-772-0416; **MINNESOTA: Minneapolis (Hopkins, 55343),** 101 Shady Oak Rd., Phone: 612-938-6371; **MISSOURI: St. Louis (Bridgeton), 63044,** 3794 McKelvey Rd., Phone: 314-291-1850; **NEBRASKA: Omaha, 68134,** 9207 Maple St., Phone: 402-391-2071; **NEW JERSEY: Fair Lawn, 07410,** 35-07 Broadway (Rte. 4), Phone: 201-791-6935; **Ocean, 07712,** 1013 State Hwy. 35, Phone: 201-775-1231; **NEW YORK: Buffalo (Amherst, 14226),** 3476 Sheridan Dr., Phone: 716-835-3090; **Jericho, Long Island, 11753,** 15 Jericho Turnpike, Phone: 516-334-8181; **Rochester, 14623,** 937 Jefferson Rd., Phone: 716-244-5470; **White Plains (North White Plains, 10603),** 7 Reservoir Rd., Phone: 914-761-7690; **OHIO: Cincinnati (Woodlawn, 45215),** 10133 Springfield Pike, Phone: 513-771-8850; **Cleveland, 44129,** 5444 Pearl Rd., Phone: 216-886-2590; **Columbus, 43229,** 2500 Morse Rd., Phone: 614-475-7200; **Toledo, 43615,** 48 S. Byrne Rd., Phone: 419-537-1887; **PENNSYLVANIA: Philadelphia, 19149,** 6318 Roosevelt Blvd., Phone: 215-288-0180; **Frazer (Chester Co.), 19355,** 630 Lancaster Pike (Rt. 30), Phone: 215-647-5555; **Pittsburgh, 15235,** 3482 Wm. Penn Hwy., Phone: 412-824-3564; **RHODE ISLAND: Providence (Warwick, 02886),** 558 Greenwich Ave., Phone: 401-738-5150; **TEXAS: Dallas, 75201,** 2715 Ross Ave., Phone: 214-826-4053; **Houston, 77027,** 3705 Westheimer, Phone: 713-623-2090; **VIRGINIA: Alexandria, 22303,** 6201 Richmond Hwy., Phone: 703-765-5515; **Norfolk (Virginia Beach, 23455),** 1055 Independence Blvd., Phone: 804-460-0997; **WASHINGTON: Seattle, 98121,** 2221 Third Ave., Phone: 206-682-2172; **WISCONSIN: Milwaukee, 53216,** 5215 W. Fond du Lac, Phone: 414-873-8250.

# This 8-bit machine, by itself, is as versatile as a lot of systems that include peripherals




Skeptical? For starters, because of its unique design the H8 is the only machine in its price class that offers full system integration, yet, with just 4K of memory and using only its "intelligent" front panel for I/O, may be operated completely without peripherals!




In addition, by using the features of its built-in Pam-8 ROM panel control program, the H8 actually allows you to dig in and examine machine level circuitry.

Responding to simple instructions the "intelligent" panel displays memory and register contents, lets you inspect and alter them even during operation. And for greater understanding, the front panel permits you to execute programs a single instruction at a time. The result is a powerful, flexible learning tool that actually lets you "see" and confirm each detail of H8's inner workings.

If you need further evidence, consider the fact that H8's system orientation allows you an almost unlimited opportunity for growth.

Memory is fully expandable, the 8080A CPU extremely versatile, and with the addition of high speed serial and parallel interfacing you gain the added flexibility of I/O operation with tape, CRT consoles, paper tape reader/punches, and soon floppy disk systems!

The H8 offers superior documentation including complete step-by-step assembly and operation manuals, is backed by 54 years of Heath reliability, and comes complete with BASIC, assembler, editor, and debug software — others charge over $60 for!

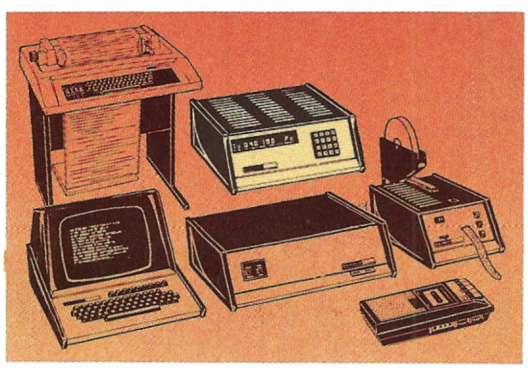
H8, simplicity for the beginner, sophistication for the expert and at $375* just right for you.

*Prices are mail order net FOB, Benton Harbor, Michigan. Prices and specifications subject to change without notice.

**HEATHKIT COMPUTERS**

**System Engineered for Personal Computing**

# H8



Be sure to use coupon on facing page of this magazine to order your FREE Heathkit Catalog!